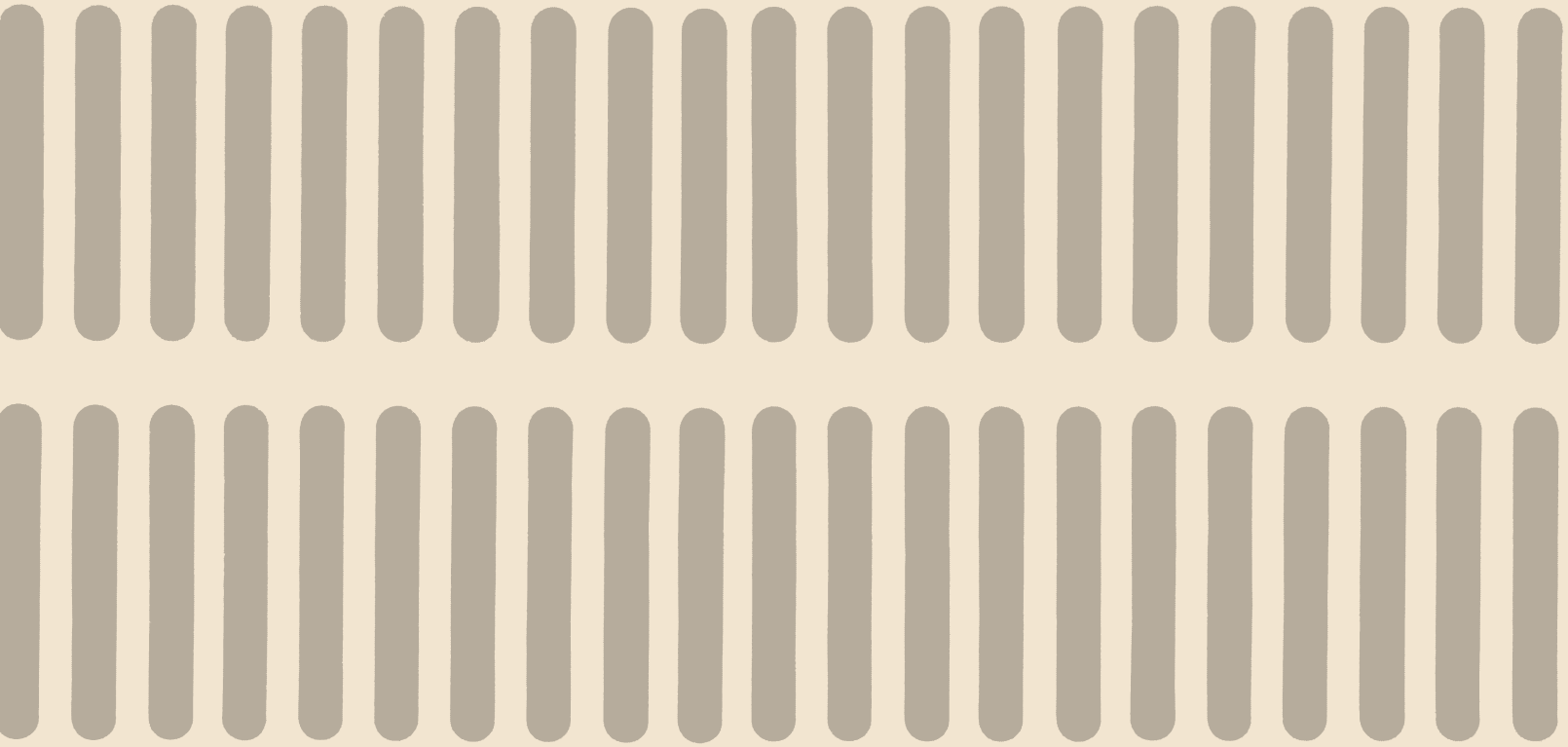
VAX11/750

Microdiagnostic Reference Manual



digital

BLANK

EK-750YA-RM-001

VAX11/750

Microdiagnostic Reference Manual

digital equipment corporation • maynard, massachusetts

1st Edition, March 1981

The material in this manual is for informational purposes and is subject to change without notice.

Digital Equipment Corporation assumes no responsibility for any errors which may appear in this manual.

Printed in U.S.A.

This document was set on DIGITAL's DECset-8000 computerized typesetting system.

Intel® 8085 is a registered trademark of Intel Corporation.

The following are trademarks of Digital Equipment Corporation, Maynard, Massachusetts:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECSYSTEM-20	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	RSTS
UNIBUS	VAX	RSX
	VMS	IAS

CONTENTS

CHAPTER 3 INSTALLATION

CHAPTER 4 FUNCTIONAL DESCRIPTION

CHAPTER 5 RUNNING MICRODIAGNOSTICS

CHAPTER 6 MICRODIAGNOSTIC MONITOR COMMANDS

FIGURES

TABLES

CHAPTER 1
INTRODUCTION

1.1 SCOPE

This manual describes the application, installation, and function of the VAX-11/750 Diagnostic Module (DM). The DM (DEC option KC750-YA) is a stored-program microcomputer that resides on a circuit board. It is installed in the VAX-11/750 CPU backplane, and provides microdiagnosis of the computer system. This manual shows how to use the module for troubleshooting. There is also a discussion of how the DM works.

Chapter 1 is an introduction to the manual. Also, it describes how the DM fits into DIGITAL's general diagnostic service.

Chapter 2 explains how to use the DM. This includes a discussion of customer responses to system problems, VAX console panel indicators, DM operating states, and diagnostic commands.

Chapter 3 describes installation and verification of the DM in the VAX-11/750.

Chapter 4 provides a functional description of the DM. Refer to this chapter to learn more about the DM hardware referred to in other chapters.

Chapters 5 through 8 describe the DM software used to run microdiagnostics on the VAX-11/750.

1.2 CONVENTIONS

The following syntax and dialogue conventions are used in this manual.

1.2.1 Command Syntax Conventions

COM <argument-1> <argument-2> [optional] [LITERAL]
{<argument-3>/<argument-4>}

COM	Type only the uppercase characters to enter the command.
<argument>	Angle brackets enclose information that must be entered with the command (such as an address). Each argument has a generic name in lowercase letters surrounded by angle brackets. Do not type the brackets; they only designate arguments.
[optional]	Square brackets enclose an argument that is optional. Do not type the brackets; they only designate optional arguments.
[LITERAL]	Literal arguments must be typed exactly as shown. They appear in all uppercase letters, and are enclosed by square brackets.

{<a-3>/<a-4>} Braces enclose a list of two or more arguments from which at least one must be selected. A slash separates the arguments. Do not type the braces or slashes; they clarify the choices.

{<a-3>!<a-4>} An explanation mark indicates that you must type one but not both of the arguments.

1.2.2 Dialogue Conventions

Sample dialogues are used to illustrate the commands discussed in this manual. The symbols used in these sample dialogues, the keys they represent, and the functions performed are described below.

Symbol	Key(s)	Function
CTRL/C	CTRL and C	Cancels current program execution.
CTRL/U	CTRL and U	Deletes all characters typed on current line. System prints carriage return/line feed sequence. Operator may retype the entire line.
CTRL/O	CTRL and O	Suppresses output to terminal.
CTRL/S	CTRL and S	Suspends output to terminal until operator types CTRL/Q.
CTRL/Q	CTRL and Q	Resumes output interrupted by CTRL/S.
<DEL>	DELETE	Deletes last character typed.
<RET>	RETURN	Serves as normal delimiter for all input to system.
CTRL/P	CTRL and P	Changes system state from program I/O to VAX console.
CTRL/D	CTRL and D	Changes system state from VAX console to DM console.
CTRL/R	CTRL and R	Displays current command string.
^		Indicates display of a control character on the terminal screen.

NOTE

Control characters are entered by typing the CTRL key in combination with another character. The combined code has special meaning for the system.

1.2.2.1 Boldface – In sample dialogues, **boldface** text indicates data entered by the operator. Text not in **boldface** indicates data typed by the computer.

1.3 OPTION OVERVIEW

The DM is a hex-height module that fits into slot 6 of the VAX-11/750 backplane. The module is a microprocessor-based computer. It can read and write to the main memory of the VAX processor, W-BUS, and control store address and data lines. The major benefit of the DM is that it can test the VAX CPU at the micro level. It can control execution of microdiagnostic programs on the CPU even if the only part of the CPU that works is the clock.

Microdiagnostic programs are stored on a TU58 cassette tape at the customer site. To test the CPU, the DM reads these programs from the tape and stores them in its own memory. Under DM control, the microdiagnostics direct the CPU. During diagnosis, the DM relays responses from the CPU to the console terminal.

CHAPTER 2
OPERATING PROCEDURES

2.1 SCOPE

This chapter explains how to use the VAX-11/750 Diagnostic Module (DM). There is a discussion of the front panel keyswitch and indicators. Also, there is a description of preprogrammed DM commands used for microdiagnosis of the VAX-11/750.

2.2 FRONT PANEL KEYSWITCH AND INDICATORS

In order to use the DM, a knowledge of the VAX-11/750 front panel keyswitch and indicators is necessary. The keyswitch determines various operating states, and the indicators show what the DM may do at a given moment. Figure 2-1 illustrates the front panel keyswitch and indicators. Table 2-1 describes the keyswitch positions. Table 2-2 describes the indicators that apply to the DM.

2.3 MICRODIAGNOSIS

The DM can diagnose the VAX-11/750 at the micro level in two ways. DM commands can perform particular actions, such as examining CPU memory locations or stepping through a microinstruction. DM commands can also load microdiagnostic programs into the DM in order to drive the VAX CPU control lines.

Service technicians usually run a microdiagnostic first. If that fails, they use DM commands to perform particular diagnostic actions. The following paragraphs explain how to use DM commands, including those that load and run microdiagnostics. (Chapters 5 through 8 describe in detail how to use microdiagnostics to test the VAX-11/750.)

2.3.1 Entering Diagnostic Commands

The VAX-11/750 must be powered up in the proper operating state to enter DM diagnostic commands. Power is applied to the system when the keyswitch is turned from OFF to another position. The VAX-11/750 powers up in either the VAX console state or the program I/O state, depending on the position of the front panel switches. To test the VAX-11/750 from the console terminal, turn the keyswitch to LOCAL. The DM runs an initialization routine that tests DM logic. Table 2-3 describes DM power-up self test activities.

The initialization routine causes the fault indicator to turn on and then off. If the fault indicator does not turn on, or turns on but not off, then a fault exists in the DM. The DM may continue to function despite a fault, but it should not be used if a fault is suspected.

The operating state of the system determines which VAX and DM commands it recognizes. Table 2-4 describes the available operating states. Figure 2-2 shows how to get from one state to another. DM commands that test the VAX-11/750 are recognized only if entered while the system is in the command mode of the DM console state.

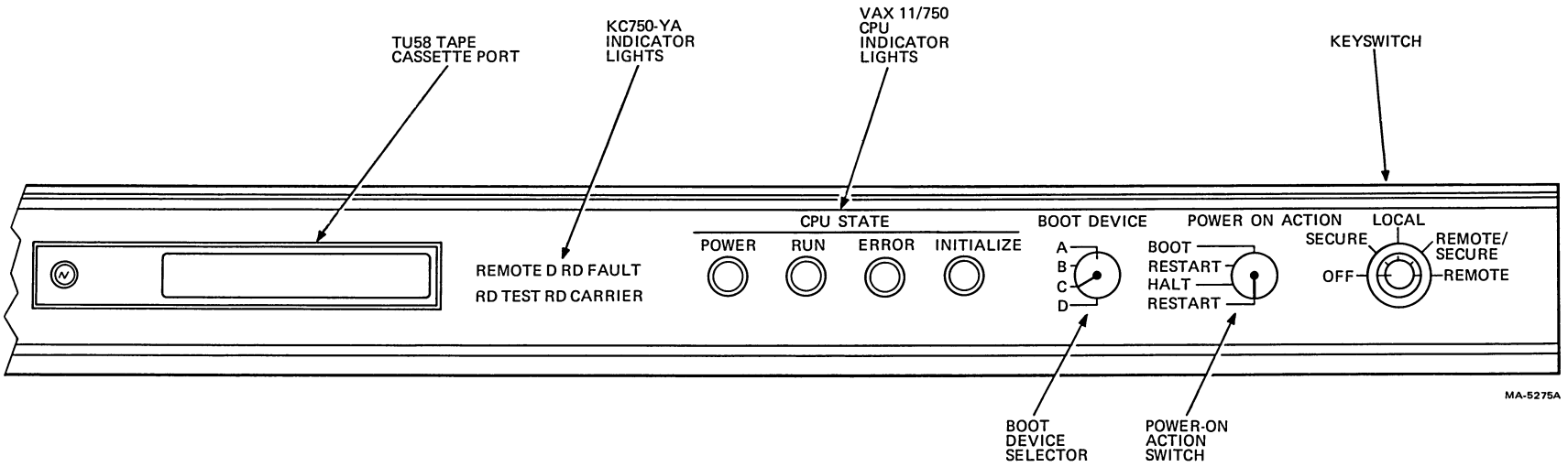


Figure 2-1 VAX-11/750 Front Panel

Table 2-1 VAX-11/750 Keyswitch Positions

Position	Description
LOCAL SECURE	System responds only to local terminal. Program I/O state enforced. This is SECURE position on keyswitch.
LOCAL	System responds only to local terminal. System responds to CTRL/D and CTRL/P to change states.
REMOTE SECURE	Same as LOCAL SECURE.
REMOTE	Same as LOCAL.

Table 2-2 DM Front Panel Indicators

Indicator	Description
REMOTE D	Not used.
REMOTE	Not used.
RD FAULT	Indicates DM logic failure. Fault indicator should turn on for about 10 seconds during console powerup as part of logic self-test.
RD TEST	Not used.
RD CARRIER	Not used.

Table 2-3 DM Power-Up Activities

Activity	Description
ROM Test	Performs a checksum calculation on the set of four DM ROM chips. Compares results to a hard-coded character located in the last position of each memory chip. Any mismatch between the calculated and hard-coded character sends an error message (ERR – ROM) to the active terminal, and the fault indicator remains on.
RAM Test	Writes, reads back, and verifies a set of bit patterns for each memory location. Any mismatch sends an error message (ERR – RAM) to the active terminal and the fault indicator remains on.

Table 2-4 System Operating States

State	Description
Program I/O	Normal operating state of the VAX-11/750. CPU is under macro program control.
VAX Console	CPU under the control of its own console microcode. It supports CPU console commands. Active terminal displays console prompt (>>>).
DM Console Control Mode	Same as program I/O state if CPU is running, except that CTRL/D changes system to DM console state command mode; running program supplies terminal prompt. If CPU is halted, this state is the same as VAX console state where VAX console prompt (>>>) is displayed.
DM Console Command Mode	System recognizes DM commands only. (Refer to Paragraph 2.3.3 for a list of available commands.) DM prompt (DM>) displayed. Character output from the CPU to the terminal is disabled.
Micro-diagnostic	DM under control of the microdiagnostic monitor (MICMON). Microdiagnostic prompt (MIC>) displayed.

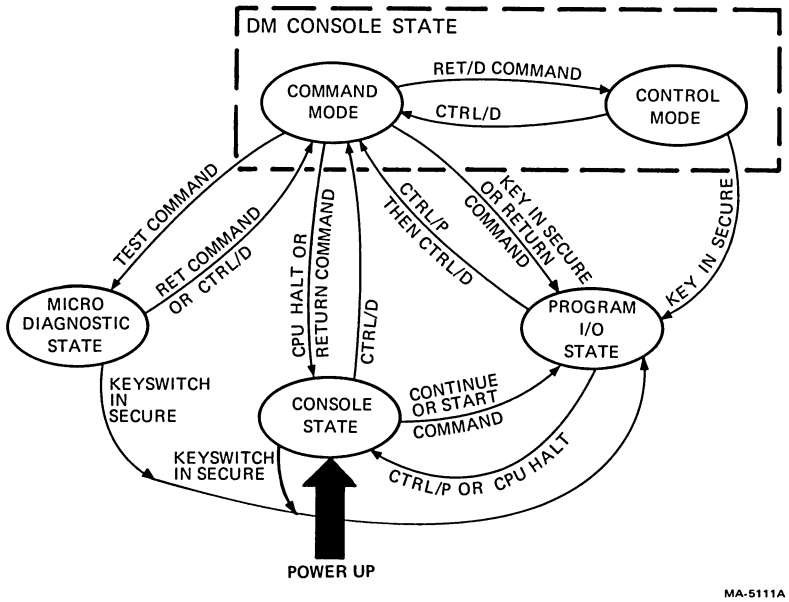


Figure 2-2 DM Operating State Transitions

To enter diagnostic commands at the console terminal, turn the keyswitch to LOCAL. If the CPU *is running* (under program control), type CTRL/P followed immediately by CTRL/D. This shifts the system from the program I/O state to the command mode of the DM console state. If the CPU *is not running,* the system is in the VAX console state; type a CTRL/D to invoke the command mode of the DM console state. Next, type the desired diagnostic command.

NOTE

If an LA120 terminal is used with the DM, turn off the auto-disconnect feature. Otherwise, CTRL/D will not invoke the command mode of the DM console state.

2.3.1.1 Example – Assume that the keyswitch is in LOCAL and CPU is not running, but the CPU clock is working. The operator wants to trace CPU control store addresses. The dialogue below shows that the operator shifts the system from the console to the DM console state. Then, the operator gives diagnostic commands to stop the CPU clock and perform a trace.

>>> **CTRL/D** (>>> is console state prompt.)

DM>**STO<RET>** (DM> is DM console state prompt.)

DM>**TR<RET>**

2.3.2 Preprogrammed Commands

Table 2-5 lists the DM preprogrammed commands. The following paragraphs describe the commands in alphabetical order.

2.3.2.1 Clear

Purpose: Clears stop-on-micromatch function.

Syntax: CL

This command disables the stop-on-micromatch function (enabled by Set command), but does not change the contents of the match register. This allows the operator to sync test equipment on the match without stopping the micromachine. Refer to Appendix A for backplane test points.

Sample Dialogue:

DM>**CL<RET>**

DM>

2.3.2.2 Deposit

Purpose: Deposits data to a VAX memory location.

Syntax: D [/modifier] <address> <data>

This command writes hexadecimal data to a hexadecimal VAX memory address specified in the command string. Modifiers (/W for word, /L for long word, and /B for byte) define the data type to be deposited. If a modifier is used, it becomes the default data type; if not, then the current default data type is assumed.

Table 2-5 DM Command Set

Command	Syntax	Function
Clear	CL	Clears stop-on-micromatch.
Deposit	D [/modifier][address]<data>	Deposits to VAX memory location.
Examine	E [/modifier][address]	Examines VAX memory location.
Examine-Console	E/C [address]	Examines DM status registers.
Initialize	INI	Initializes.
Link	LIN	Enters link control file.
Load	LO <file name.ext>[address]	Loads TU58 file to VAX memory.
Microaddress	UA <address>	Loads CS address bus.
Microaddress/C	UA/C <address>	Loads CS address bus until next M clock only.
Parity	PAR <address>	Runs control store parity check.
Perform	PER	Performs link control files.
Repeat-Last-Command	REP	Repeats console command.
Repeat-Next-Command	R <command>	Repeats following command.
Return	RET	Returns to program I/O state.
Return/D	RET/D	Returns to DM control mode.
Set	SE [address]	Sets stop-on-micromatch.
Show	SH	Shows CPU state.
Show-Version	SH/V	Shows current version of DM firmware.
Step	STE	Steps through single micro-instruction.
Step-Tick	STE/T	Steps through single clock tick.
Stop	STO	Stops clock.
Test	TE	Loads and runs microdiagnostics.
Test-Com	TE/C	Loads micromonitor and awaits command.
Test-File	TE <file name.ext>	Loads and runs user DM program.
Trace	TR	Displays trace of CS address.

If a /N modifier is used, the address field does not appear in the command. The last location where data was deposited is incremented by the default data type and used as the default address.

The address field may be replaced by an asterisk (*) or a plus sign (+). An asterisk causes the DM to use the address referenced in the last Examine or Deposit command. A plus sign (+) increments the address by one (as in the /N modifier).

The same data may be deposited to blocks of successive addresses by using the Repeat command syntax (for example, R D + <data>).

Sample Dialogue 1:

DM>**D/L 0 11223344<RET>** Long word 11223344 deposited to address zero.

DM>

Sample Dialogue 2:

DM>**D/W 2 6677<RET>**	Word 6677 deposited to address 2.
DM>**E/L 0 <RET>**	Long word examined at location 0.
P 000000 66773344	P indicates physical address.
DM>	

Sample Dialogue 3:

DM>**D/B 3 88<RET>**	Byte 88 deposited to address 3.
DM>**E/L 0<RET>**	Long word examined at location 0.
P 000000 88773344	
DM>	

Sample Dialogue 4:

DM>**D 0 10101010<RET>**	Long word 10101010 deposited to address 0.
DM>**E *<RET>**	Long word examined at address 0.
P 000000 10101010	
DM>**D/B * 11<RET>**	Byte 11 deposited to address 0.
DM>**D + 22<RET>**	Byte 22 deposited to address 1.
DM>**D + 33<RET>**	Byte 33 deposited to address 2.
DM>**D + 44<RET>**	Byte 44 deposited to address 3.
DM>**E/L 0<RET>**	Long word examined at address 0.
P 000000 44332211	
DM>	

2.3.2.3 Examine

Purpose: Examines data at a VAX memory address.

Syntax: E [/modifier] [address]

This command reads data located at a hexadecimal VAX memory address specified in the command string. Modifiers (/W for word, /L for long word, and /B for byte) define the data type to be examined. If a modifier is used, it becomes the default data type; if not, then the current default data type is assumed.

If the address field in the command is omitted, the last memory location examined is incremented by the specified or default data type and used as the default address. The address field may be replaced by an asterisk (*) in order to use the address referenced in the last Examine or Deposit command.

Blocks of addresses may be examined by using the Repeat command syntax (for example, R E).

Sample Dialogue 1:

DM>**E/L 0<RET>**	Long word examined at memory location 0.
P 000000 12345678	P indicates physical address. 000000 is address examined, and 12345678 is data examined.
DM>	

Sample Dialogue 2:

DM>**E<RET>**	Address incremented by default data type (long word) and examined.
P 000004 9ABCDEF0	
DM>	

Sample Dialogue 3:

DM>**E/W 0<RET>**	Word examined at address 0.
P 000000 5678	
DM>	

Sample Dialogue 4:

DM>**E<RET>**	Address incremented by default data type (word) and examined.
P 000002 1234	

Sample Dialogue 5:

DM>**E/B 1<RET>**	Byte examined at address 1.
P 00001 56	

DM>**E *<RET>** Byte examined at address 1.

P 000001 56

DM>

2.3.2.4 Examine Console

Purpose: Inspects data at DM random access memory (RAM) or control register address.

Syntax: E/C [address]

This command displays the contents of the DM status registers and RAM addresses. If an address is not specified, the current default address is incremented and examined. The default address is the one used in the previous Examine Console command. When using this command, data may be inspected only at addresses within the hexadecimal range 8000 to 8FF8.

Sample Dialogue:

DM>**E/C F820<RET>** DM status register F820 examined.

R F820 DF R indicates DM RAM or status register.

DM> F820 is address examined, and DF is contents of examined register in hexadecimal.

2.3.2.5 Initialize

Purpose: Initializes CPU.

Syntax: INI

This command simulates a power-fail sequence in order to recover the CPU from a hung condition. The command asserts the CPU's ACLO signal, followed five milliseconds later by DCLO. This sequence initializes the CPU to a newly powered-up state; however, to ensure the presence of a good error correction code (ECC), main memory is not swept clean.

Sample dialogue 2 shows how the initialize command may be used with Repeat Next Command to generate a scope loop of power-fail activity. This might help to deal with CPU power-fail and recovery problems.

Sample Dialogue 1:

DM>**INI<RET>**

DM>

Sample Dialogue 2:

DM>**R INI<RET>**

DM>

2.3.2.6 Link

Purpose: builds control files in DM RAM that can be executed.

Syntax: LIN

This command builds a list of instructions in DM RAM that can be executed with the Perform command. After each instruction is typed into the link list, the DM prompts the next instruction with LNK>. When there are no further instructions to enter in the link list, type a CTRL/C in response to the LNK> prompt. If the last link list instruction is Perform, the link program will loop continuously until CTRL/C is typed.

NOTE
A link list is destroyed by powering down or by entering any of the following commands: Return, Return/D, Test, Test Command, or Test File.

Sample Dialogue:

Refer to the sample dialogue provided in Paragraph 2.3.2.11 (Perform).

2.3.2.7 Load

Purpose: loads files from TU58 tape to VAX main memory.

Syntax: LO <file name.extension> [address]

This command causes the DM to locate a specific file on the TU58, read it from tape, and deposit it at sequential addresses in VAX main memory. The specified address is the starting address in VAX memory of the deposited file. If an address is not specified, the default address is zero. The file name must have exactly six characters. The extension must have exactly three characters.

The Load command loads files from the TU58 when the CPU cannot bootstrap them. This happens when the CPU is not functional enough to bootstrap, or the desired file is not properly hooked to an RT-11 boot block. In either case, the DM needs only the CPU base clock to achieve the load.

This command may be used to load diagnostics such as EVKAA, in order to get useful error information about a failing CPU.

Sample Dialogue 1:

DM>**LO ECKAL.EXE<RET>**	Diagnostic ECKAL loaded into VAX main memory starting at address 0.
DM>	

Sample Dialogue 2:

DM>**LO EVKAA.EXE 10000<RET>**	Diagnostic EVKAA loaded into VAX main memory starting at address 10000.
DM>	

Sample Dialogue 3:

DM>**LO ECSAA.EXE FE00<RET>**	Diagnostic ECSAA loaded into VAX main memory starting at address FE00.
DM>	

2.3.2.8 Microaddress

Purpose: Latches contents of selected microaddress into CPU control store latches.

Syntax: UA <address>

This command halts the CPU in order to load a selected address on the control store (CS) address bus. When the CPU restarts, the new address takes control and the CPU checks the parity of the addressed data.

NOTE
This command disables any stop-on-micromatch function previously set.

This command can perform the following functions.

1. It applies the contents of an address with bad parity to the CPU logic checkers. You may then find the bad bits statically with a scope, logic probe, or other test equipment.

2. It performs a scope loop of a given microaddress at DM execution speed. This is done by specifying Repeat execution (Paragraph 2.3.2.13) of the command and typing CTRL/O to suppress terminal output.

3. It starts the micromachine at a desired address.

NOTE
Use the Continue command to restart the CPU at the new address.

Sample Dialogue:

```
DM>UA 17FD<RET>

        CLK STOPPED CSAD 17FD        NEXT 0020

DM>
```

2.3.2.9 Microaddress/C

Purpose: Temporarily loads a specific microaddress on the control store (CS) address bus.

Syntax: UA/C <address>

This command places a selected address on the CS address lines until the next master (M) clock tick. At that point, the CPU latches data on the CS bus prior to the command. This command, therefore, does not change the program flow in the CPU.

NOTE
This command disables any stop-on-micromatch function previously set.

This command can isolate a failure in the CPU CS latching mechanism to one of the following areas.

- CS bus
- Control store
- CS latch
- CS control signal

Sample Dialogue:

DM>**UA/C 17FD**<**RET**>

DM>

2.3.2.10 Parity

Purpose: Checks VAX control store (CS) parity.

Syntax: PAR <address>

This command runs a parity check of the CPU control store. The parity check starts at the selected address and ends at the first parity error. If no parity error is found, the parity check stops at preset error address 17FD. The command displays the location of any parity error found. The CPU clock must be stopped before using this command.

Sample Dialogue:

DM>**PAR 0**<**RET**>	CPU stopped and parity check run from address 0.
PARITY ERROR CSAD 0AD1	Parity error found at address 0AD1.
DM>	

NOTE

The writable control store (WCS) may be included in the parity check by specifying hexadecimal addresses 2000 through 2400. The WCS must be loaded after power-up to avoid apparent errors resulting from power-up. The WCS is an option on the VAX-11/750.

2.3.2.11 Perform

Purpose: Executes a link program.

Syntax: PER

This command executes the program in the link file until the program stops or CTRL/C is typed.

Sample Dialogue:

DM>**LIN**<**RET**>	User builds a link program that deposits
LNK>**D/B 0 11**<**RET**>	data to successive

```
                                    bytes in the
LNK>D + 22<RET>                     long word
                                    at address 0,
LNK>D + 33<RET>                     and then
                                    examines the long
LNK>D + 44<RET>                     word at address
                                    0. The last instruction
LNK>E/L 0<RET>                      is Perform, to make
                                    the program loop
LNK>PER<RET>                        until a CTRL/C is
                                    typed.
LNK>CTRL/C                          CTRL/C terminates building
                                    of link list.

DM>PER<RET>                         Link list performed
                                    continuously until
        D/B 0 11                    CTRL/C.

        D + 22

        D + 33

        D + 44

        E/L 0

        P 000000 44332211           P indicates physical address.

        PER

        D/B 0 11

        D + 22

        D + 33

        D + 44

        E/L 0

        CTRL/C

        P 000000 44332211

        ^C

    DM>
```

2.3.2.12 Repeat Last Command

Purpose: Repeats execution of the last DM command.

Syntax: REP

This command continuously executes the preceding command until CTRL/C is typed. It helps generate scope signals off the continuous execution of selected commands. When using Repeat, type CTRL/O to stop terminal output. Repeat also helps examine or deposit to blocks of memory area. (See Paragraphs 2.3.2.4 and 2.3.2.5.)

Sample Dialogue:

DM>**E 0<RET>**	Location 0 examined.
P 000000 12345678	P indicates physical address.
DM>**REP<RET>**	Location 0 examined continuously until CTRL/C typed.
P 000000 12345678 P 000000 12345678	
CTRL/C ^C	
DM>	

2.3.2.13 Repeat Next Command

Purpose: Repeats execution of selected DM command.

Syntax: R <command>

This command continuously executes a given command until CTRL/C is typed. It helps generate scope signals off the continuous execution of selected commands. When using Repeat, type CTRL/O to stop terminal output. Repeat also helps examine or deposit to blocks of memory area. (See Paragraphs 2.3.2.4 and 2.3.2.5.)

Sample Dialogue:

DM>**R E 0<RET>**	Location 0 continuously examined until CTRL/C typed.
P 000000 12345678 P 000000 12345678 P 000000 12345678	P indicates physical address.
CTRL/C ^C	
DM>	

2.3.2.14 Return

Purpose: Returns system to program I/O state.

Syntax: RET

This command switches the system from DM console state to program I/O state.

NOTE
This command disables any stop-on-micromatch function previously set.

Sample Dialogue:

DM>**RET<RET>**
$

2.3.2.15 Return/D

Purpose: Returns system to the control mode of DM console state.

Syntax: RET/D

This command switches the system from the command mode to the control mode of DM console state. In control mode, the system is at the prompt level of the program currently running in the CPU. However, unlike program I/O state, a CTRL/D returns the system to the DM console command mode whether the CPU is running or not. Any previously set stop-on-micromatch function remains enabled after the RET/D command.

Sample Dialogue:

DM>**RET/D<RET>**

$

2.3.2.16 Set

Purpose: Enables stop-on-micromatch function.

Syntax: SE [address]

This command stops the CPU clock when the contents of the control store (CS) address bus equal the contents of the DM match register. The address to match can be specified by entering an address in the command syntax. If an address is omitted in the command syntax, the CPU stops at the address which matches the one already contained in the match register. When the CPU stops, the DM displays the addresses of both the last instruction executed and the next instruction to execute.

The command stops the CPU at a particular address from which trace of the VAX control store can be done. This shows what path the CPU took to reach that point in the control store.

Sample Dialogue:

DM>**SE 3FE<RET>**

CPU STOPPED CSAD 03FE NEXT 03FF

DM>

2.3.2.17 Show

Purpose: Displays current CPU state.

Syntax: SH

This command displays the current operating state of the CPU: running, halted, or stopped. Running means the CPU is executing code. Halted means the CPU is not executing code, but its clock is running. Stopped means the CPU clock is stopped. The command also displays the addresses of both the last instruction executed and the next instruction to execute.

Sample Dialogue:

```
DM>SH<RET>

    CPU STOPPED CSAD 3402          NEXT 3403

DM>
```

2.3.2.18 Show Version

Purpose: Displays version and date of DM firmware.

Syntax: SH/V

This command displays the current revision level and date of the firmware running in the DM. If your procedure depends on the version of the DM you have, ask the DM for the version number.

Sample Dialogue:

```
DM>SH/V<RET>

10-OCT-80 11/750 MAINT. TOOL LV = 22

DM>
```

2.3.2.19 Step

Purpose: Steps CPU through single microinstruction.

Syntax: STE

This command causes the CPU to execute a single microinstruction and stop. It then displays the address (CSAD) of the current microinstruction latched in the CPU and the address of the next microinstruction to latch.

Sample Dialogue:

```
DM>STE<RET>

    CSAD 1F02          NEXT 1F03

DM>
```

2.3.2.20 Step Tick

Purpose: Advances CPU in base (B) clock increments.

Syntax: STE/T

This command advances the CPU by one B clock tick, instead of by one master (M) clock tick as in the Step command. Step Tick can be used by itself, or with Repeat Next Command after stopping the CPU clock.

Step Tick displays the address (CSAD) whose contents are currently latched in and control the CPU. It also displays the address of the data the CPU will latch on the next M clock tick. This command allows inspection of CPU activity within a single M clock (or CPU latching) cycle.

The relationship between the B clock and M clock of the CPU is as follows. One M tick normally occurs for every three B ticks. A different action occurs on each of the three B ticks to prepare for the next M tick.

On the first B tick after an M tick, the eight HI NEXT bits in the control store address <13:06> are terminated to read back as ones. CSAD bit 13 is a special case. It is jumpered to ground on the backplane and read back as zero unless the grounding jumper has been removed (for example, after WCS installation).

On the second B tick, the microinstruction located at the NEXT address moves to the CS bus.

On the third B tick (the next M tick), the CPU latches the contents of the CS bus and the cycle begins again.

Note in sample dialogue 2 below that it takes three B ticks for a new address (IFE9) to move on and off the CS address bus.

Step Tick can diagnose problems in the VAX-11/750 system that are synchronized with the system base clock. In combination with other DM commands, a position can be established within the microexecution of a function and then advance through the execution of that function in B clock increments.

Sample Dialogue 1:

```
DM>STE/T<RET>

    CSAD 17DB           NEXT IFE9

DM>
```

Sample Dialogue 2:

```
DM>R STE/T<RET>

    CSAD 17DB           NEXT 1FE9
    CSAD 1FE9           NEXT 17DF
    CSAD 1FE9           NEXT 17DF
    CSAD 17DF           NEXT 1FEA
    CSAD 17DF           NEXT 1FEA

CTRL/C ^C

DM>
```

2.3.2.21 Stop

Purpose: Stops CPU clock.

Syntax: STO

This command stops the CPU clock, then displays the addresses of both the last instruction executed and the next instruction to execute.

Sample Dialogue:

DM>**STO<RET>**

CPU STOPPED CSAD 2F13 NEXT 2F14

DM>

2.3.2.22 Test

Purpose: Loads and runs microdiagnostics.

Syntax: TE

This command loads the microdiagnostic monitor into DM RAM, and the monitor runs a series of microdiagnostics on the VAX-11/750 system. The monitor (MICMON) and microdiagnostics reside on TU58 tape cassettes. The right cassette must be inserted before using Test.

MICMON cancels the tests if it detects an error or CTRL/C is typed while the tests are running. The monitor then prints an error message and gives the MIC> prompt on a new line. The MIC> prompt indicates that communication with MICMON in the microdiagnostic state.

If CTRL/C is typed while the DM is loading MICMON into RAM, the DM stops loading MICMON and returns to DM console command mode.

If tests have not run to completion, perform one of the following procedures to exit the microdiagnostic state.

- Type RE to exit MICMON.
- Turn the keyswitch to LOCAL SECURE or REMOTE SECURE in order to return to the program I/O state.
- Type CTRL/D to return to DM console command mode.

Refer to Chapter 5 for information and sample dialogues on how to apply the Test command.

2.3.2.23 Test Command

Purpose: Loads microdiagnostic monitor.

Syntax: TE/C

This command loads the microdiagnostic monitor (MICMON) into DM RAM, and gives the monitor control of the DM. With Test Command (unlike Test), MICMON waits for terminal commands before running any microdiagnostics. With MICMON in RAM, the system enters the microdiagnostic state and the active terminal displays the MIC> prompt.

MICMON and the microdiagnostics reside on TU58 tape cassettes. The right cassette must be inserted before using Test Command.

If CTRL/C is typed while the DM is loading MICMON into RAM, the DM stops loading MICMON and returns to DM console command mode.

To exit the microdiagnostic state, perform one of the following procedures.

- Type RE
- Turn the keyswitch to LOCAL SECURE or REMOTE SECURE in order to return to the program I/O state.
- Type CTRL/D to return to DM console command mode.

Refer to Chapter 6 for information and sample dialogues on how to apply Test Command.

2.3.2.24 Test File

Purpose: loads user program into DM RAM and runs it.

Syntax: TE <file name.extension>

This command loads programs other than the microdiagnostic monitor into DM RAM and executes them. The file name in the command argument must consist of exactly six characters. The extension must consist of exactly three characters.

Sample Dialogue:

DM>**TE MICMON.TST<RET>**

DM>

2.3.2.25 Trace

Purpose: displays the 65 most current control store (CS) addresses.

Syntax: TR

This command displays in reverse order the CS addresses stored in the DM diagnostic control store (DCS). Except in the microdiagnostic state, the DM stores the addresses of the 64 most recently executed VAX CS instructions in its DCS. The address of the next instruction to execute is stored in a DCS trap register and is also displayed.

Type CTRL/C or CTRL/D to stop the display of the 65 stored addresses at any point. Either command returns the DM> prompt at once.

The CPU clock must be stopped before using the Trace command. Trace can be used to see what path the CPU took just before its clock stopped.

Sample Dialogue:

```
DM>TR<RET>

                CSAD 2391       NEXT IF2F
                CSAD 4234
                CSAD 3178
     CTRL/C     CSAD 13^C

DM>
```

2.3.3 Error Messages

The terminal prints appropriate error messages for a variety of conditions. Error messages print on the line following the command string that caused the error condition.

With some exceptions, error messages consist of six-character codes, where the message ERR- prints, followed by the code. Table 2-6 lists these error codes.

Example

```
DM>TES<RET>                 Command to load and run microdiagnostics mistyped.
ERR – SYNTAX ERROR

DM>
```

Table 2-6 Error Message Codes

Code	Definition
TAP:14	Tape – read length error, not all records fit
TAP:13	Tape – flag received, not command or data
TAP:12	Tape – directory error
DM:11	Invalid operation code in macro
DM:10	Operation already in progress
TRM:0D	Terminal – length of input longer than buffer
TRM:0B	Terminal – command input buffer overloaded
TAP:09	Tape – file not found
TAP:08	Tape – invalid packet received
TAP:07	Tape – no end packet, invalid operation code received
TAP:06	Tape – tape count byte received exceeds maximum
TAP:05	Tape – tape check sum error received
	NOTE **UARTs are DM resident.**
TAP:04	Tape UART – overflow received
TAP:03	Tape UART – data set ready dropped
TAP:02	Tape UART – error received from UART
TAP:01	Tape UART – device timed out
CPU:04	CPU UART – overflow received
CPU:03	CPU UART – data set ready dropped
CPU:02	CPU UART – error received from UART
CPU:01	CPU UART – device timed out
TRM:04	Terminal UART – overflow received
TRM:03	Terminal UART – data set ready dropped
TRM:02	Terminal UART – error received from UART
TRM:01	Terminal UART – device timed out
TAP:FF	Tape – diagnostic failure
TAP:EE	Tape – partial operation (end of medium)
TAP:F8	Tape – bad unit number
TAP:F7	Tape – no cartridge
TAP:F5	Tape – write protocol
TAP:EF	Tape – data check error
TAP:EO	Tape – see error (block not found)
TAP:DF	Tape – motor stopped
TAP:DO	Tape – bad operation code
TAP:C9	Tape – bad record number
SYNTAX ERROR	Error in entering console commands
INVALID COMMAND	DM does not recognize command
CMI:nn	Error in VAX main memory (nn is error code); results from EXAMINE if area addressed has error
CMI:00	Nonexistent memory
CMI:01	Corrected read data
CMI:02	Read data substitute
ROM	ROM failed DM power-up selftest
RAM	RAM failed DM power-up selftest

BLANK

CHAPTER 3
INSTALLATION

3.1 SCOPE
This chapter describes how to unpack, install, and inspect the VAX-11/750 Diagnostic Module and modem.

3.2 UNPACKING AND INSPECTION
Open the shipping container carefully. Inspect all parts for damage. Immediately report any damage to the responsible carrier and branch office supervisor. Do not start installation if any item is missing or damaged; wait until the item is replaced or repaired.

3.3 INSTALLATION PROCEDURES
Perform the following procedure to install the diagnostic module (DM).

1. Power down the system by turning the keyswitch to OFF.

2. Install the L0006-YA module in slot 6 of the processor cabinet. Refer to Figure 3-1 for module placement in front of cabinet.

3. Move all cables on the processor backplane, except the TU58 power cable, from the left side of slot 6 to the right side, keeping the same vertical placement. The TU58 power cable remains in place. Refer to Figure 3-2 for cable placement in cabinet.

3.4 INSTALLATION TESTS
The following tests verify that the DM and CPU are working together properly.

- Test setup
- VAX CPU test
- VAX memory bus test
- VAX control store parity check
- Microbreak point and trace
- Microdiagnostic run test

The sample dialogues in the following paragraphs show the system responses that indicate a particular test has been passed. If responses are different from those shown here, first check to see that the test was performed correctly. If it was, then the DM is faulty and should not be used.

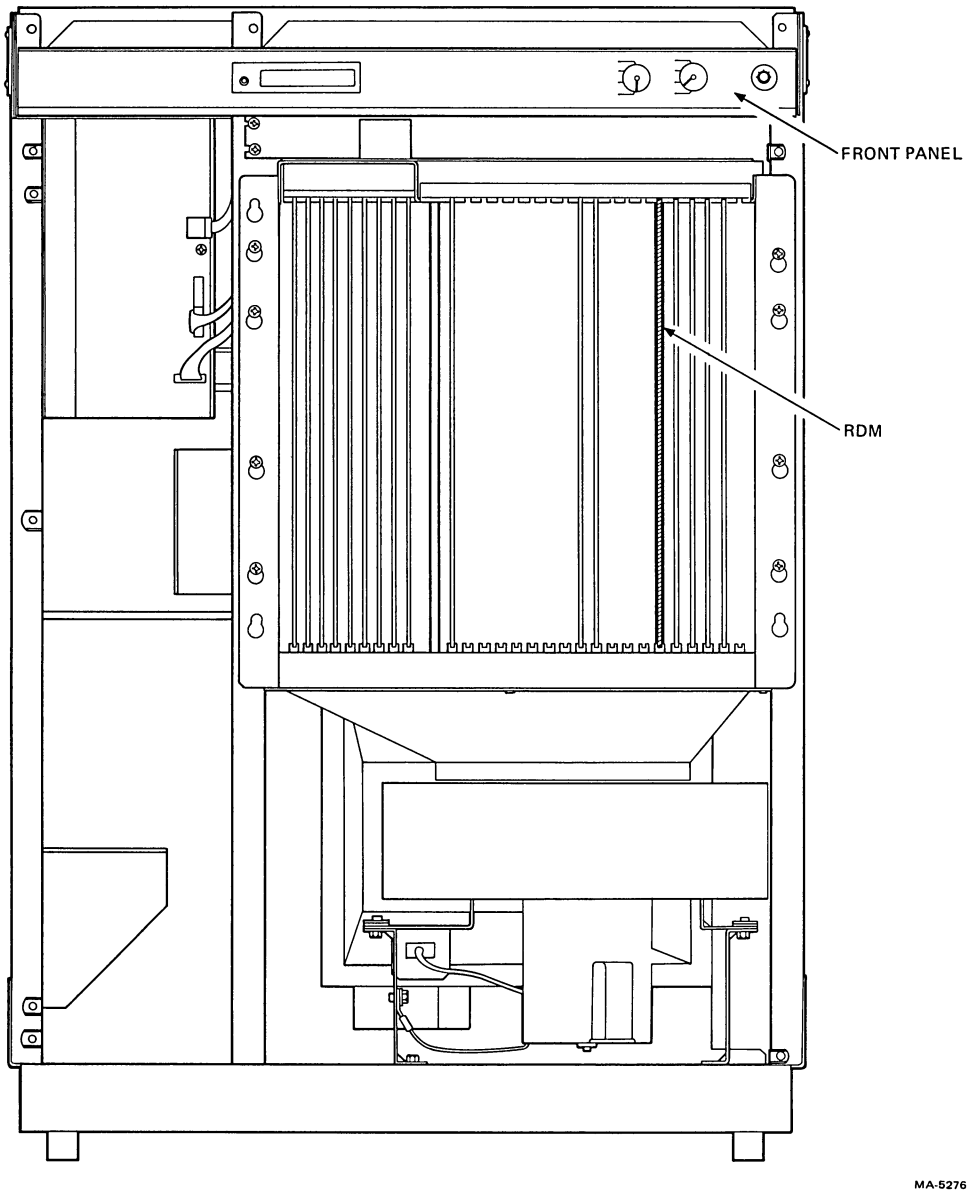


Figure 3-1 DM in VAX-11/750 Cabinet (Front View)

3.4.1 Test Setup

After installation, prepare the system for testing as follows.

1. Turn the front panel Boot Device switch to A.
2. Turn the Power On Action switch to HALT.
3. Turn the keyswitch to LOCAL.

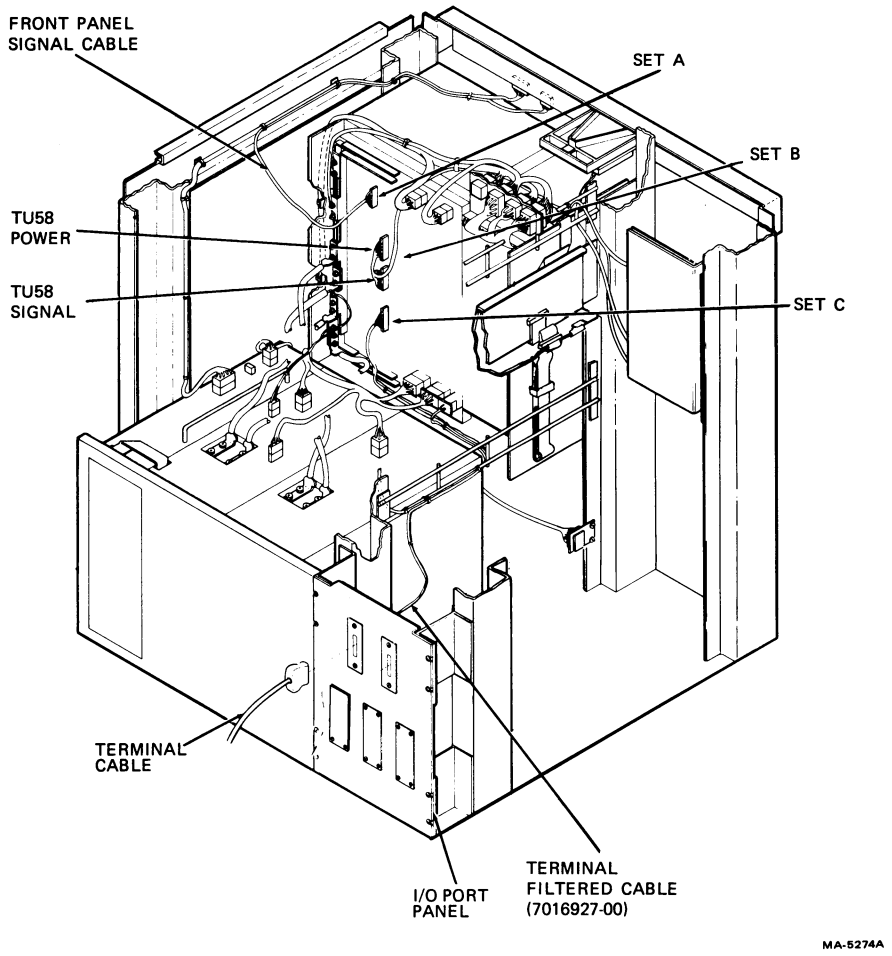


Figure 3-2 DM Cabling in VAX-11/750 Cabinet (Rear View)

The keyswitch powers up the system. The fault light should turn on, and in about ten seconds turn off. The console terminal then prints the following message.

```
% %

00000000 16
>>>
```

Any of the following conditions might indicate a faulty DM.

1. Fault light does not turn on.
2. Fault light does not turn off.
3. There is no printout.
4. Error messages containing the codes SYS, ROM, or RAM are printed.

If any of these conditions occur, turn the keyswitch to OFF. Visually inspect the DM installation. If it is correct, remove the faulty DM. Then, either install another DM or return the backplane cables to their original positions.

If no error conditions occur, perform the remaining installation tests.

3.4.2 VAX CPU Test

This test verifies CPU operation after the DM has been installed.

To perform the test, mount the TU58 tape which contains the EVKAA diagnostic, and type the following.

B<RET>

The red light on the TU58 tape drive should either blink or remain on for about two minutes while EVKAA is loaded. The diagnostic then runs repeatedly until it is stopped with a CTRL/P. The following dialogue summarizes the procedure.

```
>>>B<RET>
%%

EVKAA - 4.0 done!
EVKAA - 4.0 done!
CTRL/P
0000 nnnn 02
>>>
```

In this dialogue, n stands for some number. The EVKAA diagnostic should run at least twice before being stopped. (CTRL/P may have to be typed more than once to stop EVKAA.)

Next, use VAX firmware to write and read test data to VAX memory locations. Type the following to load memory.

```
>>>D/P/L 0 0<RET>
>>>D + FFFFFFFF<RET>
>>>D + AAAAAAAA<RET>
>>>D + 55555555<RET>
>>>D 2FFFC 12345678<RET>
>>>D + 87654321<RET>
```

Type the following to examine memory.

```
>>>E 0<RET>
        P               00000000        00000000
>>>E <RET>
        P               00000004        FFFFFFFF
>>>E <RET>
        P               00000008        AAAAAAAA
>>>E <RET>
        P               0000000C        55555555
>>>E 2FFFC<RET>
        P               0002FFFC        12345678
>>>E <RET>
        P               00030000        87654321
```

If the examined data does not agree with the deposited data, the test has failed.

3.4.3 VAX Memory Bus Test

This test verifies the ability of DM hardware to write and read data to VAX memory. (Perform the VAX CPU test before this test.)

To perform the test, first enter DM console state.

```
>>>CTRL/D
DM>
```

Examine VAX memory.

```
DM>E 0<RET>
P  000000  00000000
DM>E<RET>
P  000004  FFFFFFFF
DM>E<RET>
P  000008  AAAAAAAA
DM>E<RET>
P  00000C  55555555
DM>E 2FFFC<RET>
P  02FFFC  12345678
DM>E<RET>
P  030000  87654321
```

Deposit to memory.

```
DM>D 0 FFFFFFFF<RET>
DM>D + 0<RET>
DM>D 2FFFC 55AA55AA<RET>
DM>D + AA55AA55<RET>
```

Reexamine memory.

```
DM>E 0<RET>
P  000000  FFFFFFFF
DM>E<RET>
P  000004  00000000
DM>E 2FFFC<RET>
P  02FFFC  55AA55AA
DM>E<RET>
P  030000  AA55AA55
DM>
```

If the data read from memory is not the same as data written into memory, the test has failed.

3.4.4 VAX Control Store Parity Check

This test exercises the DM's ability to know whether the VAX clock is running, to stop the VAX clock, and to check the VAX control store parity. (Perform the VAX memory bus test before this test.)

If the DM is working properly, it refuses to check VAX control store parity while the VAX clock is running. Thus, in order to see if the DM can detect the VAX clock, ask it to do a parity check while the VAX clock is running.

```
DM>PAR<RET>
CLK RUNNING
DM>
```

Check if the DM can stop the VAX clock.

```
DM>STO<RET>
     CLK STOPPED CSAD 096n      NEXT 096n
```

In this dialogue n stands for some number.

Now, test the DM's ability to check the parity of the VAX control store. (An error has been preset at microaddress 17FD.)

```
DM>PAR 0<RET>
PARITY ERROR CSAD 17FD
DM>
```

If any other address appears in the dialogue, a parity error has been discovered at that address in the VAX control store.

3.4.5 Microbreak Point and Trace

This test exercises the DM logic that stops the VAX clock when the CPU executes the microinstruction at a preset microaddress (called the microbreak point). The test also exercises the DM's ability to trace VAX control store addresses. A trace is a listing (in order of execution) of the CS addresses of the last 65 microinstructions executed. (Perform the VAX control store parity check test before this test.)

First, initialize the CPU.

```
DM>RET<RET>
% %

00000000 16

>>>CTRL/D

DM>
```

Set the microbreak point.

```
DM>SE A2B<RET>
DM>
```

The microbreak point set happens to be the address of the CPU's carriage return microinstruction. Thus, when the CPU does a carriage return, the DM should stop the CPU clock.

Return to the VAX console state. In the process, the CPU does a carriage return.

```
DM>RET/D<RET>
CLK STOPPED CSAD 0A2B        NEXT nnnn
```

Where n is some hexadecimal number and NEXT indicates the address of the next instruction to be executed.

Now, check if the DM does a VAX control store trace.

DM>**TR<RET>**

```
CSAD        0A2B        NEXT nnnn
CSAD        nnnn
CSAD        nnnn
....        ....
CSAD        0965
CSAD        0964
CSAD        0966
CSAD        0964
CSAD        0966
....        ....
```

In all, 65 microaddresses should be displayed. In the above dialogue, an ellipsis (....) shows a continuation of address printouts. As before, n represents some hexadecimal number and NEXT is the address of the next microinstruction to be executed.

Note that the address of the last microinstruction executed (0A2B) is the address of the carriage return microinstruction. The addresses 0964 and 0966 are the loop in which the CPU waits for a carriage return.

3.4.6 Microdiagnostic Run Test

The most DM hardware is tested when a microdiagnostic is run on the VAX. Before beginning this test, make sure the system is in the DM console state. The microdiagnostic used here is the data path module (DPM) microdiagnostic. Find the TU58 tape with this program, and mount it in the VAX front panel. Use the following dialogue to load the microdiagnostic into DM memory and run it twice.

DM>**TE/C<RET>**

MIC>**DI PA:2<RET>**

```
ECKAA-V02.00 DPM-V02.00
01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B, 0C, 0D,
0E, 0F, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A,
1B, 1C, 1D, 1E, 1F, 20, 21, 22, 23,
END OF PASS 01

DPM-V02.00
01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B, 0C, 0D,
0E, 0F, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A,
1B, 1C, 1D, 1E, 1F, 20, 21, 22, 23,
END OF PASS 02
```

MIC>**RET<RET>**

DM>

Note that the MIC> prompt is displayed when the system is in the microdiagnostic state.

If the printout is not the same as the one given above, and the test has been performed correctly, replace the DM with a spare. If there is no spare DM, remove this DM and return the backplane cables to their original positions.

3.5 REMOVAL PROCEDURES

The DM is removed by reversing the installation procedure (Paragraph 3.3.).

WARNING

Always power down the system before installing or removing the DM.

CHAPTER 4
FUNCTIONAL DESCRIPTION

4.1 SCOPE

This chapter provides information about the internal operations of the VAX-11/750 Diagnostic Module (DM). It describes the general DM functions, its major components, and how they work together. While reading this chapter, refer to Figure 4-1 (the DM system block diagram).

4.2 GENERAL FUNCTIONS

The following list describes the general functions of the diagnostic module.

1. Allows users to perform all regular processor console functions from the console terminal.
2. Loads and reads VAX-11/750 main memory.
3. Loads and reads processor W-BUS in support of microdiagnostics.
4. Drives VAX-11/750 control lines in place of regular control store during microdiagnostics.
5. Traces and stores addresses of executed VAX-11/750 control store instructions.

NOTE
The following functions cannot be performed at the same time.

6. Compares the addresses of instructions executed by the main processor with a preselected address, and stops the processor when a match occurs (at the microbreak point).
7. Runs a parity check on the processor control store.
8. Conducts logic selftest of the DM on powerup.

4.3 MAJOR COMPONENTS

The major DM components are a preprogrammed memory, a microprocessor, a diagnostic control store, and miscellaneous registers and control logic.

4.3.1 Preprogrammed Memory

Preprogrammed memory consists of three parts: software switches, logic self-tester, and program loader. Some DM preprogrammed memory functions connect to preprogrammed memory functions in the VAX-11/750 console. Therefore, the following discussion includes appropriate VAX console functions to complete the descriptions.

4.3.1.1 Software Switches – Software switches are instructions that change the DM state and perform certain control functions. Each DM state corresponds to a particular position of the software switches (Figures 4-2 and 4-3).

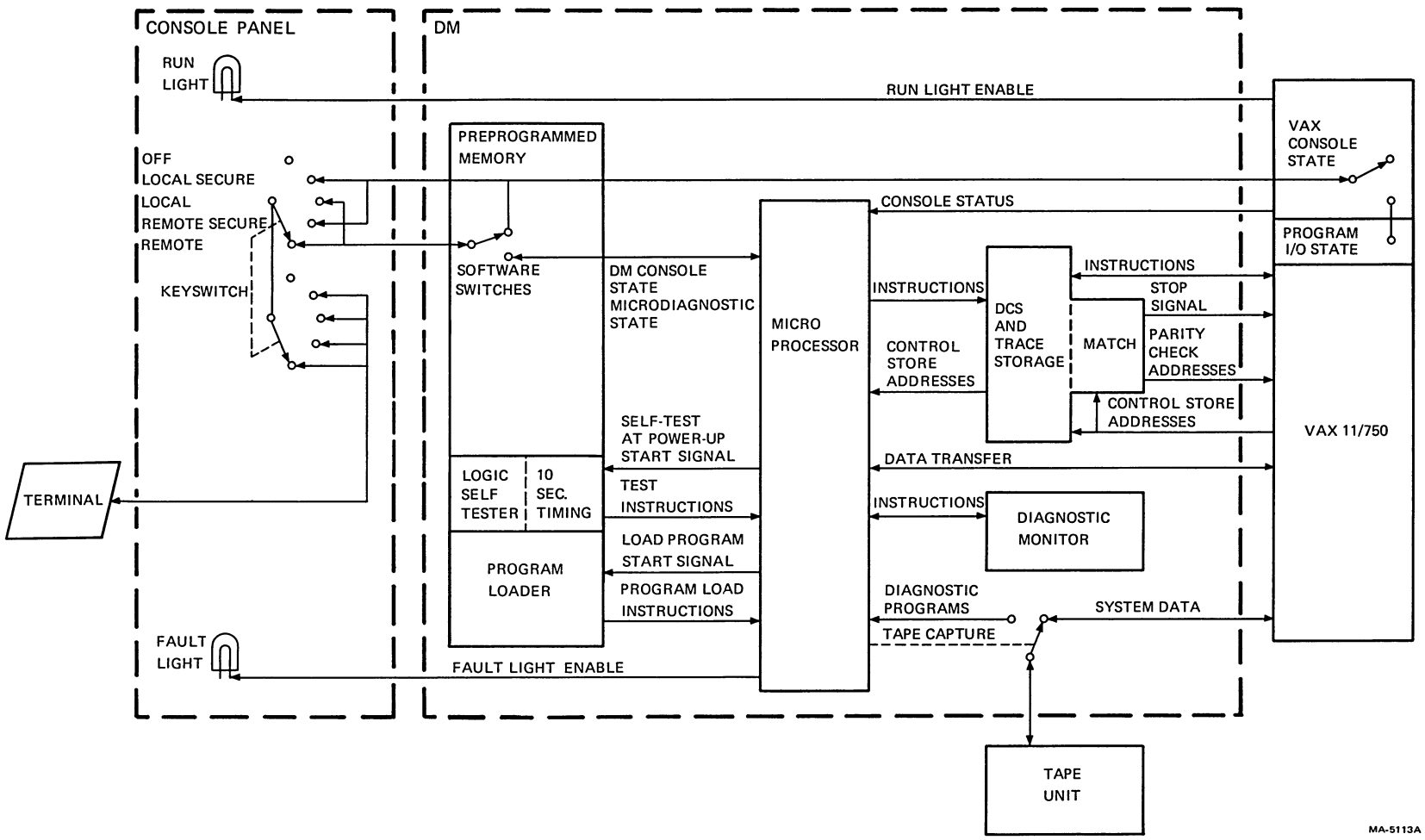


Figure 4-1 DM System Block Diagram

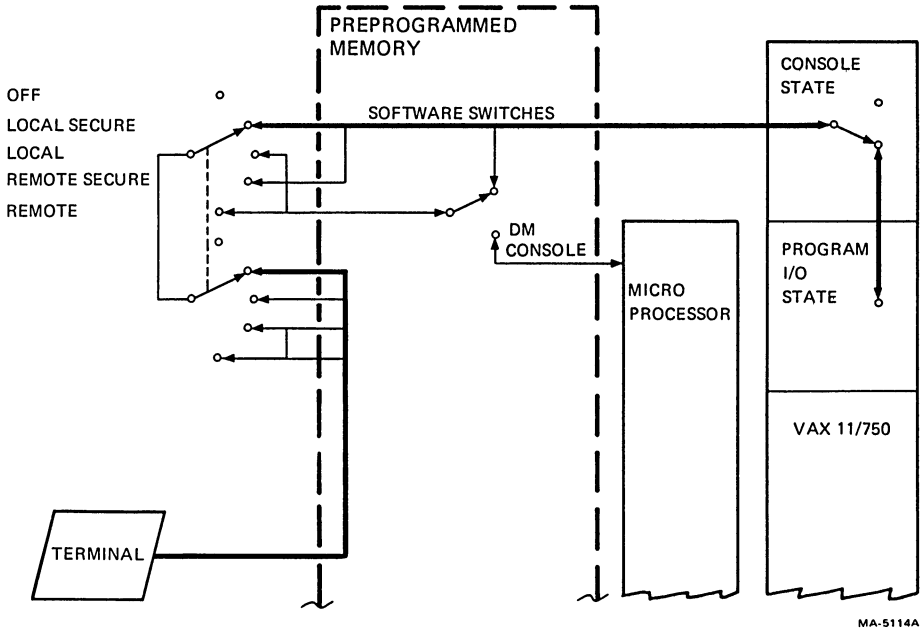


Figure 4-2 Local Secure Program I/O State

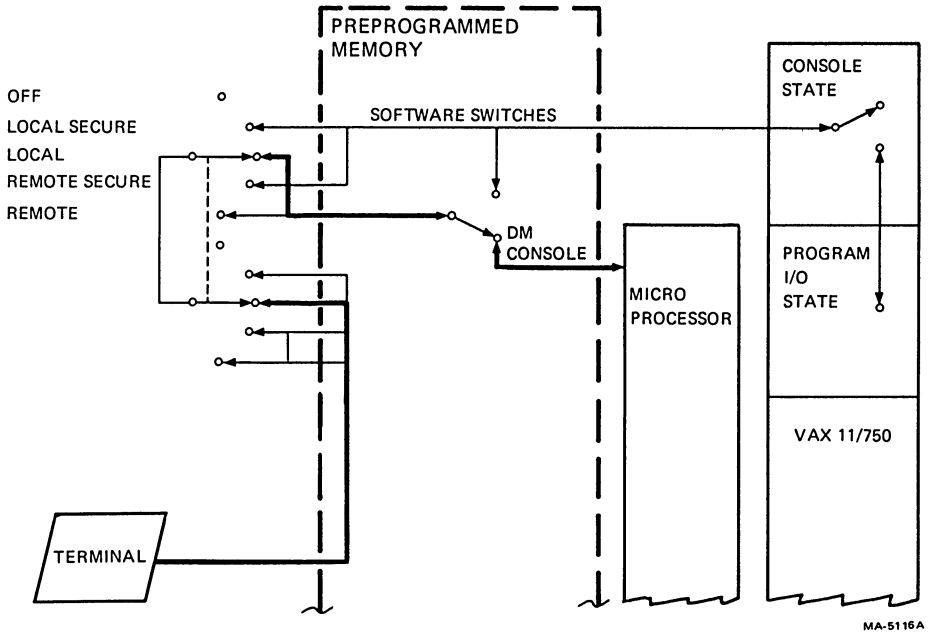


Figure 4-3 Local DM Console State

In program I/O state, the console terminal communicates with the VAX-11/750 at the program level. In VAX console state, the terminal communicates with VAX-11/750 firmware. The VAX console and DM monitor these lines for control characters that set software switches to change states. In program I/O state or VAX console state, the DM's microprocessor plays no active role in relaying information to the VAX-11/750. Switching between these two states is the exclusive job of the VAX console.

The DM microprocessor accepts commands from the console terminal only after the system switches to DM console state. The DM does this switching. The DM processor only communicates directly with the terminal when the system is in DM console state. The DM console state (and other states entered via the DM console state, such as the microdiagnostic state) may only be entered from the terminal if the keyswitch is not in the SECURE position.

Chapter 2 describes all commands that switch system states and perform other control functions. Preprogrammed memory supports the control commands. The VAX console, DM console, or both may recognize a given command. The current system state determines whether VAX or DM supports a command. For example, VAX supports CTRL/O in either the program I/O or VAX console states; DM supports CTRL/O in the DM console or microdiagnostic states.

4.3.1.2 Logic Self-Tester – The logic self-tester is a group of preprogrammed instructions that verify DM logic. On DM powerup, the microprocessor signals the self-tester for instructions. These instructions tell the microprocessor to check various hardware parts and turn on the fault indicator. If the microprocessor does not find a hardware failure within 10 seconds, the microprocessor turns the fault indicator off. The logic self-tester runs only at powerup.

4.3.1.3 Program Loader – The program loader is a group of preprogrammed instructions called by the microprocessor to load the microdiagnostic monitor from the tape unit into the DM's memory. (The role of the microdiagnostic monitor is discussed in Paragraph 4.3.2.2.) There is no hard connection between the tape unit and DM. The DM has no knowledge of VAX's use of the tape unit. The DM has a logical connection to the tape unit that the microprocessor uses to gain control of the tape unit from the main processor.

4.3.2 Microprocessor

The microprocessor (an Intel 8085) executes instructions in the preprogrammed memory and microdiagnostic monitor. Since the DM has its own processor, preprogrammed memory, and stored program capacity, the microprocessor does not depend on the VAX processor to run tests. Only the VAX clock must be operating.

4.3.2.1 Data Transfer Paths – The microprocessor uses a data transfer line to read and write to two important areas in the VAX processor: W-BUS and main memory. The W-BUS transfers information between the VAX-11/750's arithmetic logic unit (ALU) and general registers. Main memory stores programs and data to be used later. The DM tests for problems by writing and then reading known data to these areas. (The diagnostic module's WD register and MD register hold data to be written to the W-BUS and memory, respectively. The module's WH register and MH register hold data read from the W-BUS and memory, respectively. The module's MA register holds the address of memory that has been read from or written to.)

4.3.2.2 Microdiagnostic Monitor – The microdiagnostic monitor is the DM's nonresident operating system. It directs microdiagnosis of the VAX-11/750. To load the monitor, the operator enters the Test command. The program loader instructs the microprocessor to load the monitor from the tape unit into an area of random access memory (RAM).

The Test command switches the DM to the microdiagnostic state. (Before this happens, the keyswitch and software switch connections must be made, so that the system is in DM console state.) Once in the microdiagnostic state, commands entered on the active terminal address the diagnostic monitor which controls the microprocessor.

4.3.3 Diagnostic Control Store

The following paragraphs discuss the function of the diagnostic control store and its associated hardware.

4.3.3.1 Diagnostic Program Storage – The diagnostic control store (DCS) is the area of memory where the microprocessor loads specific diagnostic routines. Once loaded, these diagnostic programs drive the VAX-11/750, in effect replacing the VAX control store. The microdiagnostic monitor loads the DCS and initiates driving of the VAX processor.

4.3.3.2 Trace Storage – Trace is another function that occupies the DCS. The trace function continually records the 64 most recently executed control store addresses, as well as the address of the current microinstruction for the VAX to execute. Trace storage shares part of the memory used for storing diagnostic programs, so the two functions cannot be performed at the same time.

If the VAX processor stops, tracing can determine why. The microprocessor can relay control store addresses stored in memory to the terminal for analysis.

In trace mode, the DM continually and automatically clocks CS addresses from the CS address lines into a trap register, and from there into 64 DCS RAM locations. The DM is in trace mode whenever it is not in the microdiagnostic state.

When the DM loads a new CS address into the trap register, the previous CS address in the trap register moves into a DCS RAM location. The value of a DCS pointer determines the DCS RAM location to which the CS address moves. Every time an address moves from the trap register to RAM, the pointer increments by one. The DM loads the trap register when the CPU latches a word.

The DCS pointer has a capacity of 64. After 64 words have been loaded into RAM, the pointer increments to the location loaded first. The DCS RAM, therefore, is a circular buffer in which new CS addresses from the trap register overwrite the oldest information.

Chapter 2 describes the Trace command. That command displays a readout of the DCS. Reading out of the DCS is the reverse process of writing into the DCS. The microprocessor reads CS addresses out of the DCS and forwards them to the terminal for display.

The first address read is the address in the trap register, which is always the latest CS address. The next address is read from the current RAM location specified by the pointer. After each DCS location is read, the pointer decrements by one, until all locations have been read.

4.3.3.3 Match Register – In parallel with the DCS, a match register also monitors VAX CS address lines. The match register has four functions.

1. Stop-on-micromatch
2. Generating a known test pulse
3. Gaining control of the VAX control lines for the DCS
4. VAX control store parity checking

Stop-on-micromatch is a ROM-encoded function that loads the match register with a specific control store address. It then compares the address with that of each control store instruction executed. When the address of the current control store instruction matches the selected address, the match logic stops the processor. Primarily, stop-on-micromatch permits you to trace the path the processor took to a particular microinstruction.

When the address in the match register matches the address on the CS address lines, a pulse generates on pin C0681 of the CPU backplane. By deliberately making the CPU cycle on a matching address, the match signal can be used as a known pulse to trigger test equipment.

In addition, the match register helps the DCS gain control of the VAX processor. Under microdiagnostic monitor control, the microprocessor loads the match register with a DCS address. The match register places this DCS address on the control store address lines to force control of the VAX to the DCS. Once the DCS has control, the particular microdiagnostic in the DCS drives the VAX processor control lines. The microdiagnostic is then responsible for making the CPU continue to select DCS as a source of instructions.

The match register also helps to check parity in the VAX control store. The microprocessor can make the match register increment through successive control store addresses. At each location, the VAX-11/750's own parity check logic is enabled to perform a parity check. The match register addresses each control store location until the register increments to an address in the CPU control store that holds a deliberately set parity error. (If the parity check uncovers an accidental parity error, the match register does not increment past that address.) Detecting the known parity error is a test of the VAX-11/750 parity check logic.

NOTE

When parity checking the writable control store (WCS), first load the WCS with data. WCS addresses are 2000 to 23FF.

CHAPTER 5
RUNNING MICRODIAGNOSTICS

5.1 SCOPE

The VAX-11/750 microdiagnostic program consists of two basic blocks: microdiagnostic monitor and test overlays. When the microdiagnostic program starts, a ROM-based TU58 driver routine loads the microdiagnostic monitor (MICMON) from the TU58 tape cartridge into 8085 RAM memory. The monitor remains in the 8085 memory for the duration of the test session.

Once the monitor is loaded, the program initialization routine begins. When several passes are run, this routine executes at the beginning of each. The program initialization routine tracks program flow, initializes parameters, and invokes the TU58 driver routine to load the first test overlay for execution. Normally, the last instruction in each test overlay invokes the TU58 driver routine to load the next test overlay. The test overlays are read into memory one at a time.

Test overlays generally consist of pseudo-instructions, test data, and microinstructions to execute from the diagnostic control store (DCS). The monitor reads the pseudo-instructions and invokes appropriate routines. These routines implement the required functions in 8085 code. The pseudo-instructions set up and control the execution of microinstructions in DCS.

The monitor may be used to set and clear flags, and issue other commands to control the program flow with precision.

5.2 RUNNING THE MICRODIAGNOSTIC PROGRAM

DIGITAL distributes the VAX-11/750 microdiagnostic program on a series of TU58 tape cartridges. Each tape cartridge contains the microdiagnostic monitor and a number of test overlays. The microdiagnostic monitor is identical on all tape cartridges, but the test overlays are unique. Each tape cartridge tests a portion of the CPU or memory. Normally, you should run the tapes in the following order: DPM (ECKAB), MIC (ECKAC), FPA (ECKPE).

To run the microdiagnostic program, perform the following procedure.

1. Insert the desired TU58 cartridge into the slot on the CPU front panel.

2. Turn the Power On Action switch to HALT.

3. Turn the keyswitch to LOCAL to power up the CPU. If the console terminal executes microverify successfully, it should type two percent signs, and then the console prompt >>>.

4. Type CTRL/D to invoke the DM console command mode. The console terminal should respond with the DM prompt DM>.

5. Type TE to run the microdiagnostic program. If the test sequence on the tape cartridge runs to completion without error, the program types a series of messages on the console terminal confirming completion. The following example shows a typical printout with no errors.

Sample Dialogue:

```
DM>TE<RET>                                  ! Test command.

ECKAA-V0.22 MIC (L0003)-V00.04              ! Program
                                            !  identification.

01,02,03,04,05,06,07,08,09,0A,0B,0C,0D,     ! Test numbers typed
0E,0F,10,11,12,13,14,15,16,17,18,19,1A,     ! at execution time.

1B,1C,1D,1E,1F,20,21,22,23,                 ! Number of tests
END OF PASS 01                              ! varies depending
                                            ! on tape.

MIC>                                        ! Microdiagnostic
                                            !  monitor prompt.
```

6. Replace the tape cartridge with the next microdiagnostic tape and type DI.

7. Type RE<CR> after the MICMON prompt to return to the console I/O mode.

8. Type RET<CR> after the DM command mode prompt to return to the VAX console state.

5.3 MICRODIAGNOSTIC MESSAGES

The microdiagnostic program prints messages on the console terminal to indicate the following items and functions.

1. Monitor name, version, and module under test
2. Test number to execute next (current test)
3. Error messages

If the program detects an error, it halts test execution, types an error message, and passes control to the microdiagnostic monitor. Refer to the following example.

Sample Dialogue:

```
DM>TE<RET>

ECKAA-V0.22 MIC (L0003)-V00.04              ! Program identification.
01,                                         ! Test number.
?ERROR: 0030 TEST: 01 SUBTEST: 01

DATA:   AAAAAAAA
        AAAA8AAA                            ! MODULE NAME is listed only
        00000001                            ! if it is not the module
                                            ! being tested.

FAILING GATE ARRAYS: ADK,

MIC>                                        ! Microdiagnostic monitor
                                            ! prompt.
```

The only test number printed is 01, indicating that the program started test 01 but did not reach test 02. The next line in the message confirms the location of the failure in test 01.

```
?ERROR: 0030 TEST: 01 SUBTEST: 01
```

The 0030 entry indicates the test program counter (PC) value of the failing pseudo instruction. The data patterns indicate the expected and received data and the loop count. If the program identifies failing gate arrays, they are listed at this point. The message then lists the names of other modules that may be causing the failure.

In most cases, the machine should be repaired by turning off power, replacing the module or gate arrays called out, and rerunning the microdiagnostic program. If the message calls out several modules, replace them one at a time in the order listed and run the program after each replacement until the fault is repaired.

Replacing the parts called out may not repair the fault. Also, spare modules may not be available. In such cases, you may want to examine the program listing for the failing test. You may then use microdiagnostic monitor commands to set up scope loops and step through the failing test. The following chapters cover these topics.

BLANK

CHAPTER 6
MICRODIAGNOSTIC MONITOR COMMANDS

6.1 SCOPE

Several commands control the microdiagnostic monitor. The monitor may be entered either directly or indirectly. To enter directly from the DM console command mode, type TE/C (Test command) instead of TE (Test). To enter directly during test execution, type CTRL/C. The microdiagnostic program enters the monitor indirectly (automatically) if it finds either a TU58 driver error or a hardware error when the halt on error flag is set. When the monitor is waiting for a command, it prompts with MIC>.

The list below contains legal command op codes.

Diagnose
Loop
Set
Clear
Show
Return
Continue

The list below contains legal command keyword arguments.

TEST
VBUS
LOOP
STEP
PASS
NER (no error report)
CYCLE (one microcycle)
BELL
TICK (one-half microcycle)
FLAG
HALT
SA (signature analyzer)
SOMM (stop-on-micromatch)
INSTRUCTION
CONTINUE
CF (control file)
IB (inhibit burst)
QA (quality assurance)
TR (trace flag)
QV (quick verify)
DM (diagnostic modules)

In the following command descriptions, an exclamation point means exclusive OR. In sample dialogues and examples, an exclamation point indicates that what follows on that line is a comment.

6.2 DIAGNOSE

Syntax: DI TE:<test-number> [<test-number>] ! CO ! QV PA:<pass-count>!DM]

This command takes three types of arguments. It initializes the program control flags and starts program execution.

Flag	Initial Setting
LOOP	clear
NER	clear
BELL	clear
HALT	set
IB	clear
SA	clear
QA	clear
TR	no change

If Diagnose is typed without any arguments (switches), the monitor executes all tests on the installed tape once. Control then returns to the DM console command mode, if the program detects no errors.

Sample Dialogue:

```
MIC>DI<RET>                                    ! Diagnose command.

ECKAA-V0.21 MIC (L0003)-V00.04

01,02,03,04,05,06,07,08,09,0A,0B,0C,0D,0E,0F,10,11,12,13,
14,15,16,17,18,19,1A,1B,1C,1D,1E,1F,20,21,22,23,
MIC>
```

6.2.1 Diagnose Test

The TEST argument may be used with the Diagnose command in three ways.

1. Diagnose TEST followed by a single test number executes the specified test indefinitely. Type CTRL/C to escape from the loop and return to the monitor.

2. Diagnose TEST followed by a test number and Continue executes the specified test and all following tests on the installed tape cartridge.

3. Diagnose TEST followed by two test numbers executes all tests between and including the two numbers specified. Control then returns to the monitor. If the first test number typed is out of range, the program types an error message and returns control to the monitor.

Sample Dialogue 1:

```
MIC>DI TE:2<RET>                    ! Loop on test 2
                                    ! indefinitely.
ECKAA-V0.21 MIC (L0003)-V00.04      ! Monitor identifies itself.
                                    ! Test runs, looping
CTRL/C                              ! continuously.
02^C                                ! Test number is 2.
                                    ! CTRL/C to escape.
MIC>                                ! Microdiagnostic monitor
                                    ! prompt.
```

Sample Dialogue 2:

```
MIC>DI TE:2 CO<RET>                    ! Start with test 2 and
                                       ! run remaining tests
                                       ! on tape cartridge.
ECKAA-V0.21 MIC (L0003)-V00.04         ! Monitor identifies itself.
02,03,04,05,06,07,08,09,               ! Each test number is
0A,0B,0C,0D,0E,0F,10,11,12,13,         ! typed as that test starts.
14,15,16,17,18,19,1A,1B,1C,
1D,1E,1F,20,21,22,23,
END OF PASS 01
MIC>                                   ! Prompt.
```

Sample Dialogue 3:

```
MIC>DI TE:3 6<RET>                     ! Execute tests 3,4,5,6.
ECKAA-V0.21 MIC (L0003)-V00.04         ! Program identifies itself.
03,04,05,06,                           ! Each test number is typed
                                       ! as that test starts.
MIC>                                   ! Prompt.
```

6.2.2 Diagnose Pass

The PASS argument may be used with the Diagnose command to set the pass count to a specified number. The default pass count is 1. A pass count of 0 causes the program to run indefinitely. At the end of the specified number of passes, control returns to MICMON.

Sample Dialogue:

```
MIC>DI PA:2<RET>

ECKAA-V0.21 MIC (L0003)-V00.04

01,02,03,04,05,06,07,08,09,0A,0B,0C,0D,0E,0F,10,11,12,13,
14,15,16,17,18,19,1A,1B,1C,1D,1E,1F,20,21,22,23,
END OF PASS 01

01,02,03,04,05,06,07,08,09,0A,0B,0C,0D,0E,0F,10,11,12,13,
14,15,16,17,18,19,1A,1B,1C,1D,1E,1F,20,21,22,23,
END OF PASS 02
MIC>
```

6.2.3 Diagnose Quick Verify

The QV (quick verify) argument may be used with the Diagnose command to run a predetermined subset of tests on microdiagnostic tape. It was designed for use by DIGITAL manufacturing groups.

Sample Dialogue:

```
MIC>DI QV<RET>

ECKAA-V0.21 MIC (L0003)-V00.04
26,27,28,....
END OF PASS 01
MIC >
```

6.2.4 Diagnose Diagnostic Module

The DM (diagnostic module) argument may be used with the Diagnose command to run those tests that check out the diagnostic module only. This command should be used only if the VAX-11/750 is working properly.

Sample Dialogue:

```
MIC> DI DM<RET>
ECKAA-V0.21 RDM (L0006)-V00.04
01,02,....
END OF PASS 01

MIC>
```

6.3 SHOW FLAGS

Syntax: SH FL

This command displays the current states of program control flags listed in Table 6-1.

Sample Dialogue:

```
MIC>SH FL<RET>                          ! Show flags.

FLAGS SET: LO, NE, BE,
FLAGS CLEAR: HA, SA, IB, QA, TR

MIC>
```

6.4 SET FLAG

Syntax: SE FL <flag-name-list>

This command sets (enables) any of the program control flags.

Sample Dialogue:

```
MIC>SE FL HA                            ! Set HALT flag.
MIC>SE FL LO                            ! Set LOOP flag.
MIC>SE FL NE                            ! Set NO ERROR
                                        ! REPORT flag.
MIC>SE FL BE                            ! Set BELL flag.
MIC>SE FL SA                            ! Set SIGNATURE
                                        ! ANALYSIS flag.
MIC>SE FL QA                            ! Set QUALITY
MIC>SE FL QA:<loop-count>               ! Issues error messages on
                                        ! loop counts greater than or
                                        ! equal to <loop-count>
                                        ! ASSURANCE flag.
MIC>SE FL IB                            ! Set INHIBIT BURST flag.
MIC>SE FL LO NE BE                      ! Set LOOP, NO ERROR, and BELL
                                        ! flags.
```

Table 6-1 Program Control Flags

Flag	Description
HALT	Halt. Calls the monitor when error is detected and error message has been typed.
LOOP	Loop on error. This flag is useful only in the event of a program-detected error. Flag may be set with the Loop command, or manually by typing SE FL LO. When flag is set and program detects an error, test loops on minimum amount of code necessary to recreate the error. The loop executed may include pseudo-instructions (8085 operations) and DCS microinstructions, running from ERRLOOP instruction to IFERROR instruction in failing test. Or loop may include DCS microinstructions only. If IB flag is set or a microinstruction trap occurs, program does not loop on microinstructions in DCS. Refer to Paragraph 7.17 (BRSTCLK) and Paragraph 6.15 (Loop command). Error messages are not inhibited while loop is executing unless NER flag is set. Type CTRL/C to escape from loop and return to monitor.
NER	No error report. If flag is set, program does not report errors.
BELL	Bell on error. If flag is set, program rings terminal bell on first occurrence of an error and on every fifth occurrence.
SA	Signature analysis. If flag is set, program loops on test in progress, between BEGINSA and ENDSA pseudo-instructions. Loop occurs whether or not the test detects errors. Set SA flag when using a signature analyzer to help diagnose faults. This flag provides two sync points on the backplane: a start/stop window (slot 6, pin C75) and a clock pulse (slot 6, pin C73). With test looping and signature analyzer connected to sync points, signature analyzer analyzes any test point that probe samples. Signature analyzer displays a value (signature) if signal pattern is steady. Compare this value with corresponding value from a known good module to locate failures.
QA	Quality assurance. If flag is set, program responds to each test as if it had detected a failure.
IB	Inhibit burst flag. If this flag and the loop flag are set, the program does not try to loop on DCS microcode only. Rather, it loops between ERRLOOP and IFERROR pseudo-instructions.
TR	Trace flag. If flag is set, monitor types test names as well as test numbers.

6.5 CLEAR FLAG

Syntax: CL FL <flag-name-list>

This command clears (disables) any of the program control flags.

Sample Dialogue:

```
MIC>CL FL HA<RET>                    ! Clear HALT flag.
MIC>
```

6.6 SET STOP-ON-MICROMATCH

Syntax: SE SO:<cs-address>

This command stops execution of code in DCS at the specified address. DCS addresses range from 0 to 3F. Add 1800 to the desired DCS address. Type the Continue Command after the MIC> prompt to proceed from your current halt, after typing SE SO <CS-address> to get to that address.

Sample Dialogue:

```
MIC>SE SO:1803<RET>                          ! Stop at DCS address 3.
MIC>CO<RET>                                  ! Continue.
MICRO BREAK MATCH, CS ADDR=1803, DCS ADDR=03.
MIC>
```

The function of the Set Stop-On-Micromatch command may be deceiving at times. Suppose you want to stop microcode execution at a location that follows a microinstruction specifying the NEXT field. Assume that the following code is now in the DCS:

```
DCS_DATA

;% 00    NOP
;% 01    MEMSCAR_R[TEMP0]                    ; Physical/Virtual
                                             ; address to
                                             ; MEMSCAR.
;% 02    MEMSCR_R[TEMP1]                     ; Set MME off.
;% 03    MEMSCAR_R[TEMP2]                    ; Reference cache
                                             ; only to MEMSCAR.
;% 04    MEMSCR_R[TEMP3]                     ; Turn CMI off.
;% 05    PSL(PREV_CURM ISCURM_R[TEMP4])      ; Set mode to
                                             ; executive.
;% 06    STROBE.V.BUS,WB_RDM,PTE CHECK READ?,NEXT/1800
                                             ; Perform PTE check.
;% 07    NOP,STOP.CLOCK.LATCH.CS.ADDR        ; Stop CPU clock.
```

If this code was in the DCS, typing SE SO:1807 would not access DCS address 7. Instead, to get to DCS address 7, stop-on-micromatch must be set to stop at the address specified by the NEXT field in DCS address 6 (MIC>SE S0:1800). This is because Set Stop-on-Micromatch compares the control store address specified in the command with the address asserted on the NEXT control store lines. The stop always occurs at the next DCS address after the address whose NEXT field contains the value specified in the command.

```
MIC>SE SO:1800<RET>                          ; Stop at DCS address 7.
MIC:CO<RET>                                  ; Continue.
MICRO BREAK MATCH, CS ADDR = 1800, DCS ADDR = 07
MIC>
```

The NEXT address field never affects the DCS microcode. When DCS is active, the NEXT address field merely selects the DCS range of addresses (1800 to 183F). DCS address control resides in a counter. The counter can only increment or return to 0. A microdiagnostic test specifies the NEXT field to test the microaddress generation function, not to create a microcode jump.

6.7 CLEAR STOP-ON-MICROMATCH

Syntax: CL SO[:<cs-address>]

This command clears the stop-on-micromatch function. Add 1800 to the desired DCS address to create a scope sync pulse at that address. If <cs-address> is specified, a scope sync pulse is generated on slot

6, pin C81 when the current address matches <cs-address>. The pulse occurs with the M clock that marks the beginning of the specified microcycle.

Sample Dialogue:

```
MIC>CL SO:1803<RET>          ! Clear stop-on-micromatch
                             ! function at DCS address 3.
                             ! Load 1803 into the DM match
                             ! register.
MIC>CO<RET>                  ! Continue.
```

6.8 SET CONTROL FILE

Syntax: SE CF:<dcs-address> <bit-number>

This command sets the specified <bit-number> in the control file of the specified <dcs-address>. (Refer to Paragraph 8.6.) Table 6-2 lists the control file bit functions and bit numbers.

Sample Dialogue:

```
MIC>SE CF:3 4<RET>           ! Set control file bit
                             ! 4 (HRWBUS) in
                             ! DCS address 3.
MIC>
```

Table 6-2 Control File Bit Functions

Bit	Mnemonic	Function
0	VSTB	Strobes visibility bus at rising edge of next M clock.
1	DCSACL	Clears DCS address register.
2	STPCLK	Stops VAX-11/750 CPU clock.
3	ENTRACE	Latches trace register at rising edge of the next M clock. When this bit is not set, trace register monitors control store address lines continuously.
4	HRWBUS	Reads W-BUS by strobing WH register at rising edge of the next M clock.
5	WBUSDR	Enables WD register onto W-BUS (write W-BUS).
6	STBCMI	Reads CMI data by strobing MH register with CMI data at rising edge of next M clock.
7	SATRIG	Inhibits clocking signature analyzer on next M clock. This signal connects to a post on module for use by signature analyzer (backplane pin C73 on slot 6.)

6.9 CLEAR CONTROL FILE

Syntax: CL CF: <dcs-address> <bit-number>

This command clears the specified bits in the control file of the <dcs-address>.

Sample Dialogue:

```
MIC>CL CF:3 4<RET>          ! Clear control file bit
                            ! 4 (HRWBUS) in
                            ! DCS address 3.
```

6.10 SET STEP INSTRUCTION

Syntax: SE ST IN[:<test-pc>]

This command steps through the pseudo-instructions (8085 code) in the current test. If <test-pc> is specified, the step function does not start until the instruction at the <test-pc> address is ready to execute. If <test-pc> is not specified, the step function begins at the next pseudo-instruction of the current test, following a Loop or Continue command.

In either case, when the program stops at the next pseudo-instruction, press the space bar to continue stepping through the pseudo-instructions. Type a carriage return to escape from the step mode back to the monitor.

Sample Dialogue 1:

```
MIC>SE ST IN<RET>           ! Set step instruction.
MIC>LO<RET>                 ! Start looping.
     TPC = 001F             ! Test PC 1F.
                            ! Space bar.
     TPC = 0024
     <RET>                  ! Carriage return.
MIC>
.
.
.
```

Sample Dialogue 2:

```
MIC>SE ST IN:2C<RET>        ! Set step instruction
                            ! starting at 2C.
MIC>CO<RET>                 ! Continue
     TPC = 002C             ! Test PC is 2C.
                            ! Space bar.
     TPC = 0038
                            ! Space bar.
     TPC = 003E
     <RET>                  ! Carriage return.
MIC>
```

6.11 SET STEP CYCLE

Syntax: SE ST CY

This command steps through DCS microinstructions one CPU machine cycle at a time.

After entering in this command, the Continue, Loop, or Diagnose commands must be typed to begin stepping. After the program stops at the first DCS address in the current test, press the space bar to step through the test. Type a carriage return to escape and return control to the monitor.

Sample Dialogue:

```
MIC>SE ST CY<RET>                ! Set step cycle.
MIC>LO<RET>                      ! Loop.
     DCS ADDR = 01
                                 ! Space bar.
     DCS ADDR = 02
                                 ! Space bar.
     DCS ADDR = 03
     <RET>                       ! Carriage return.
MIC>
```

6.12 SET STEP TICK

Syntax: SE ST TI

This command steps through DCS microinstructions, stopping twice in every CPU machine cycle, as a function of the phase clock.

After entering this command, the Continue, Loop, or Diagnose commands must be typed to begin stepping. After the program stops at the first DCS address in the current test, press the space bar to step through the test. Type a carriage return to escape and return control to the monitor.

Sample Dialogue:

```
MIC>SE ST TI<RET>                ! Set step tick.
MIC>LO<RET>
     DCS ADDR = 00               ! First half of B clock.
                                 ! Space bar.
     DCS ADDR = 00               ! Second half of B clock.
                                 ! Space bar.
     DCS ADDR = 01               ! Third half of B clock.
                                 ! Space bar.
     DCS ADDR = 01               ! Fourth half of B clock.
     <RET>                       ! Carriage return.
MIC>
```

6.13 SHOW VISIBILITY BUS

Syntax: SH VB

This command strobes the visibility bus (VBUS) and prints out the current signal states. This command is most useful after executing the Initialize pseudo-instruction, and when stepping through microcode in DCS. Show Visibility Bus displays the 40 bits of the visibility bus, with the low-order bit on the right.

Sample Dialogue:

```
MIC>SH VB<RET>
      VBUS= 10101011,01000010,00110000,10110101,00111111
MIC>
```

Table 6-3 lists the signal names, bit numbers, module names, and print set sheets for the visibility bus signals.

6.14 CONTINUE

Syntax: CO

This command continues testing after the program stops, following error detection or CTRL/C. This command does not modify any flags.

Sample Dialogue:

```
MIC>DI<RET>                                    ! Run all tests.

ECKAA-V0.22 MIC (L0003)-V00.04
01,
?ERROR: 0030 TEST: 01 SUBTEST: 01              ! Error report.

 DATA:     AAAAAAAA
           AAA8AAAA
           00000001

MIC>CO<RET>                                    ! Continue.
02,03,04,05,06,07,08,09,0A,0B,0C,0D,0E,0F,10,11,12,13,14,
15,16,17,18,19,1A,1B,1C,1D,1E,1F,20,21,22,23
END OF PASS 01
MIC>
```

If you get an unexpected clock stop error message, DI must be typed to resume testing. Typing CO simply returns you to the monitor.

6.15 LOOP

Syntax: LO

This command puts the program in an error loop after it detects and reports an error. If the SA flag is set, the loop is a signature analysis loop. The command clears the HALT flag, sets the NER and LOOP flags, and performs a Continue command function.

Sample Dialogue:

```
?ERROR: 0030 TEST: 01 SUBTEST: 01

DATA:     AAAAAAAA
          AAAA8AAA                             ! Error report.
          00000001
FAILING MODULES: MIC,

MIC>SE ST CY<RET>                              ! Set step cycle.
```

Table 6-3 Visibility Bus Signals

Bit No (Hex)	Bit No (Decimal)	Signal Name	Module and Print Set Sheet No
00	0	UBI03 FORCE TB PE L	UBI15
01	1	UBI03 FORCE CACHE PE L	UBI15
02	2	CS HNEXT PAR H	UBI15
03	3	UBI03 RTUT DINH L	UBI15
04	4	UBI03 BUSF PAR H	UBI15
05	5	UBI15 INT PEND L	UBI15
06	6	UBI15 UB INT GRANT H	UBI15
07	7	UBI12 CON HALT L	UBI15
08	8	MICRO VECTOR 3 H	MIC04
09	9	MICRO VECTOR 2 H	MIC04
0A	10	MICRO VECTOR 1 H	MIC04
0B	11	MICRO VECTOR 0 H	MIC04
0C	12	MIC07 GEN DEST INH L	MIC04
0D	13	MIC07 UTRAP L	MIC04
0E	14	MIC04 LATCHED MBUS 15 L	MIC04
0F	15	MIC04 PROC INIT L	MIC04
10	16	MIC04 MSRC XB H	MIC04
11	17	MIC04 MEM STALL H	MIC04
12	18	MIC04 STATUS VALID H	MIC04
13	19	MIC05 UB REQ H	MIC04
14	20	MIC07 CORR DATA INT L	MIC04
15	21	MIC07 WR BUS ERR INT L	MIC04
16	22	Not Used, always 0	MIC04
17	23	Not Used, always 0	MIC04
18	24	DPM17 INSTR FETCH H	DPM21
19	25	DPM17 DO SRVC L	DPM21
1A	26	DPM16 IRD1 H	DPM21
1B	27	DPM14 UVCTR BRANCH H	DPM21
1C	28	DPM14 DISABLE HI NEXT H	DPM21
1D	29	DPM20 CS PARITY ERROR H	DPM21
1E	30	DPM14 LD DSR L	DPM21
1F	31	DPM17 PSL CM H	DPM21
20	32	DPM19 ISIZE 0 L	DPM21
21	33	DPM19 ISIZE 1 L	DPM21
22	34	DPM18 DST RMODE H	DPM21
23	35	DPM13 TIMER INT L	DPM21
24	36	DPM19 D SIZE 0 H	DPM21
25	37	DPM19 D SIZE 1 H	DPM21
26	38	DPM11 MCS TMP L	DPM21
27	39	UBI13 MSEQ INIT L	DPM21

```
MIC>LO<RET>                          ! Loop.
        DCS ADDR= 00                 ! First DCS address.
        DCS ADDR= 01                 ! Press space bar to step.
        DCS ADDR= 02
        DCS ADDR= 03
        DCS ADDR= 04
        DCS ADDR= 05
        DCS ADDR= 06
        DCS ADDR= 00                 ! Loop, and return to the
        DCS ADDR= 01                 ! first DCS address. See
        DCS ADDR= 02                 ! Paragraph 7.17 for
        DCS ADDR= 03                 ! conditions under which
        DCS ADDR= 04                 ! the DCS address is not
        DCS ADDR= 05                 ! reset to 0.
        <RET>                        ! Carriage return.
MIC>
```

6.16 RETURN

Syntax: RE

This command returns the monitor from the microdiagnostic program to DM console command mode.

Sample Dialogue:

```
MIC>RE<RET>                          ! Return.
DM>                                  ! DM console command mode
                                     ! prompt.
```

CHAPTER 7
PSEUDO-INSTRUCTIONS

7.1 SCOPE

In VAX-11/750 microdiagnostic programs, the monitor interprets and implements pseudo-instructions. Psuedo-instructions consist of many 8085 instructions. This chapter describes the pseudo-instructions used to diagnose the VAX-11/750.

In the pseudo-instructions described below, an unspecified or byte data type equals eight bits. A word data type is 16 bits, and a long data type equals 32 bits. A microword equals 11 bytes.

7.2 INITIALIZE

Syntax: **INITIALIZE**

This pseudo-instruction initializes the VAX-11/750 CPU as follows.

1. Cycles DC LO
2. Asserts PROCINIT (processor initialize)
3. Disables CMI
4. Disables memory management
5. Loads PSL with zeros

7.3 LOAD DCS

Syntax: LOADDCS <src-addr>, [src-index], <dcs-addr>, <wrd-cnt>

This pseudo-instruction moves <wrd-cnt> microwords from the 8085 memory buffer to DCS, starting at <dcs-addr>. <src-addr>, indexed by [src-index], specifies the starting 8085 memory address.

7.4 LOAD FIELD

Syntax: LDFIELD <src-addr>,[src-index],[data-type], <dst-addr>,
{<start-bit>, / <address-of-start-bit-table>}
{<end-bit>, / <address-of-end-bit-table>}
[longlit], [start-bit-ind], [end-bit-ind], [no-parity],
[NS]

This pseudo-instruction acts only on DM 8085 memory. It is normally used before the LOADDCS pseudo-instruction to set up a microword before moving it into DCS. LDFIELD moves a field of length (<end-bit>-<start bit>) indexed by [SRC index] (times [data-type], to a field in <dst-addr> starting at <start-bit>.

If <address-of-start-bit-table> or <address-of-end-bit-table> is specified, the field is specified by the contents of the associated table. If [start-bit-ind] or [end-bit-ind] is specified, the respective table address is indexed by bytes. If [longlit] is specified, data is inverted before being placed in the LONLIT field. If you specify [no-parity], the monitor does not calculate parity on that microword. Use the optional [NS] argument to load data unscrambled into the 8085 memory. [NS] moves data other than microwords, which must be scrambled when loaded into the DCS.

Example:

```
        .
        .
        .
LDFIELD         2$,,B,1$,0,7,
LOADDCS         1$,,0,1
1$:
;% NOP,NEXT/0                          ; Microinstruction.
2$:     .BYTE   ^X3F
```

In the previous example, the LDFIELD instruction moves 3F at 2$, to the microinstruction stored in 8085 memory at 1$. The 3F is inserted from bit 0 to bit 7. The microinstruction at 1$ becomes NOP,NEXT/3F. LOADDCS then moves this microinstruction to DCS address 0. The next subtest may use the same microinstruction, but place a different value in any field with the LDFIELD pseudo-instruction.

7.5 LOAD REGISTER

Syntax: LOADREG <reg-addr>,<src-addr>,[src-index],[data-type]

This pseudo-instruction loads a DM register from the 8085 memory. LOADREG moves the contents of <src-addr>, indexed by [src-index], to the DM register specified by <reg-addr>.

Example:

```
LOADREG         WDREG,1$,I,L

1$:             .LONG           ^XFFFFFFFF
```

In the previous example, the LOADREG instruction loads 32 bits of the WD register with ones. Assume that the index I equals 1.

7.6 LOAD INDEX

Syntax: LDINDEX <index-nam>,{<index-value>/
<index-value-add>,[index]}

This pseudo-instruction initializes the specified index, <index-nam> (I,J, or K), to the specified value, <index-value>. The instruction can also initialize the <index-nam> to <index-value-add> times [index].

7.7 SAVE INDEX

Syntax: SAVEINDEX <index-nam>,<dst-addr>

This pseudo-instruction moves the current value of the specified index, <index-nam>, into <dst-addr> in 8085 memory.

7.8 INCREMENT INDEX

Syntax: INCINDEX <index-nam>

This pseudo-instruction increments the current value of <index-nam> (I,J, or K).

7.9 STUCK AT ZERO OR ONE PATTERN

Syntax: SA01PAT <index-nam>,<dst-addr>,[longlit],[dst-addr-x]

This pseudo-instruction selects one of the following patterns each time it is executed.

P1 10101010101010101010101010101010 (B)
P2 01010101010101010101010101010101 (B)
P3 00110011001100110011001100110011 (B)
P4 00001111000011110000111100001111 (B)
P5 00000000111111110000000011111111 (B)
P6 00000000000000001111111111111111 (B)

The pseudo-instruction chooses a pattern according to the current value of <index-nam>. If [longlit] is not specified, it places the pattern in the test data region of the 8085 memory at <dst-addr>. An <index-nam> with a value of 1 selects pattern P1, and so on. The current value of <index-nam> must be less than or equal to 6.

If [longlit] is specified, the pseudo-instruction places the pattern in bits <62:31> of the microword starting at <dst-addr>. If [dst-addr-x] is specified, the pattern is also placed at this address.

7.10 BEGIN SIGNATURE ANALYZER

Syntax: BEGINSA

If the SA flag is set, this pseudo-instruction causes the program to loop between the BEGINSA and ENDSA pseudo-instructions.

7.11 END SIGNATURE ANALYZER

Syntax: ENDSA

This pseudo-instruction checks the SA flag. It marks the end of the signature analyzer loop.

If the SA flag is clear, the program executes the next pseudo instruction. (Refer to the example in Paragraph 8.3.)

If CTRL/C is typed in a SA loop, control returns to the monitor.

7.12 LOOP

Syntax: LOOP <loop-name>,<start-value>,<end-value>

This pseudo-instruction initializes loop parameters for a specified index <loop name> (I,J, or K). If the <start-value> is less than the <end-value>, the loop is an incrementing loop. If the <end-value> is less than the <start-value>, the loop is a decrementing loop. Loops of this type are used to repeat a test or subtest, using a different data pattern indexed by <loop-name> each time.

7.13 END LOOP

Syntax: ENDLOOP <loop-name>

This pseudo-instruction terminates a programmed loop. The program jumps to the pseudo-instruction following the corresponding LOOP pseudo-instruction, unless the required number of loops have been completed. A loop of this type functions whether or not an error has been detected.

7.14 ERROR LOOP

Syntax: ERRLOOP

This pseudo-instruction saves the test PC. The instruction works with the IFERROR pseudo-instruction. When the test detects an error and the LOOP flag is set, the program loops on the error.

7.15 IF ERROR

Syntax: IFERROR [data-cnt],[gate-array-list],[module-name]

This pseudo-instruction checks the results of one of the three compare pseudo-instructions: CMPREG, CMPREGMSK, and CMPVBUS.

Based on flag settings, the instruction executes a loop or continues with the program. The optional parameters specify the nature of the error messages. (Refer to the example in Paragraph 5.3.)

[Data-cnt] specifies the number of RTEMP registers to type in the error message. For example, a [data-cnt] of 4 specifies RTEMPs 0, 1, 2, 3. These registers contain test data as described in the error description in the test listings.

7.16 FETCH

Syntax: FETCH <dcs-addr>,[MIC-ADR-INH]

This pseudo-instruction initiates a DCS microinstruction sequence. It loads the microword stored at <dcs-addr> into the control store latches. If [MIC-ADR-INH] is specified, the instruction leaves the Micro Address Inhibit signal asserted, after execution of the instruction, to inhibit the processor from driving the control store lines.

7.17 BURST CLOCK

Syntax: BRSTCLK <stop-addr>, [INHIBIT]

This pseudo-instruction bursts the VAX-11/750 CPU clock under normal circumstances. It checks to see that the microcode stopped at <stop-addr>. Control then passes to the next pseudo-instruction.

If the following conditions exist, the instruction performs the functions set control file bit 1 (DCSACL) and clear control file bit 2 (STPCLK) at <stop-addr>.

1. Error occurs
2. LOOP flag set
3. <INHIBIT> argument not specified
4. SA flag clear

In this way, BRSTCLK creates a DCS loop. Otherwise, control passes to the next pseudo-instruction.

BRSTCLK also controls the step cycle, step tick, and stop-on-micromatch functions. (Refer to the Loop command description in Paragraph 6.15 and the LOOP flag description in Paragraph 6.3 for details of the looping function.)

7.18 EXPECT MICROTRAP

Syntax: EXPUTRAP

This pseudo-instruction enables the BRSTCLK function to recognize a microtrap. If a microtrap occurs as expected, the burst clock routine does not report an error and passes control to the next pseudo-instruction.

If a microtrap occurs, the control store address lines are forced outside the range of DCS address space, 1800 to 183F. The DM monitors the control store address lines, stopping the CPU clock when they are forced outside the range of DCS.

If an EXPUTRAP pseudo-instruction precedes a microtrap, the program handles the trap. Otherwise, the program prints an unexpected clock stop error message on the console terminal.

If a microtrap occurs, whether or not it is expected, you cannot create a DCS loop.

7.19 COMPARE REGISTER

Syntax: CMPREG <reg-addr>,<dst-addr>,[dst-index], [data-type], [mode-type]

This pseudo-instruction compares the contents of a DM register <reg-addr> with <dst-addr>, indexed by [dst-index] in DCS.

If the [mode-type] is blank, the CMPREG pseudo-instruction expects the data to be equal. If the [mode-type] is NE, it expects the data to be unequal.

On a comparison failure, the IFERROR pseudo-instruction following CMPREG prints an error message.

7.20 COMPARE REGISTER MASKED

Syntax: CMPREGMSK
<reg-addr>,<dst-addr>,[dst-index],<msk-addr>,
[msk-index],[data-type],[mode-type]

This pseudo-instruction works exactly like CMPREG, except that the comparison is masked by <mask-addr> and indexed by [mask-index]. Only bit positions where the mask equals 1 are compared, as shown in the following example.

Register Contents	1 1 1 1 1 1 1 0
Destination Contents	0 1 1 0 1 1 1 0
Mask	0 1 1 1 0 0 0 0

Only these three bit positions are compared.

7.21 MASK

Syntax: MASK
<src-addr>,[src-index],<msk-addr>,[msk-index],
[data-type]

This pseudo-instruction masks bits in an 8085 memory location. Bits that are ones in the mask are preserved in the location at <src-addr>.

7.22 COMPARE VISIBILITY BUS

Syntax: CMPVBUS <data-table-ptr>,[index]

This pseudo-instruction compares the values of bits on the visibility bus (VBUS) with the expected values specified in the data table. If the comparison fails, the IFERROR pseudo-instruction prints an error message.

Example

```
        BRSTCLK     7                          ! End DCS program.
        CMPVBUS     2$,I                       ! Compare VBUS results.
        IFERROR     ,<<ACV><ADK><UTR>>,        ! Possible failing arrays.

2$:     .BYTE       ^X4                        ! VBus data table.
        VBUSDATA    <^X08>,0                   ! Iteration 1.
        VBUSDATA    <^X09>,0
        VBUSDATA    <^X0A>,1
        VBUSDATA    <^X0B>,1
```

In the above example the VBUS data is compared with data in the table at 2$. .BYTE ^X4 at the top of the table indicates that the comparison involves four bits. VBUSDATA <^X08>, 0 indicates that bit 8 will be the first bit position compared, and the expected value is 0.

7.23 NEW TEST

Syntax: newtest <title>, [section], [alt-test-number]

NEWTEST is always the first pseudo-instruction in a test. It defines the beginning of the test and performs a variety of housekeeping functions.

7.24 SUBTEST

Syntax: subtest [subtest number]

This pseudo-instruction increments the current subtest number or takes the value given in the optional argument.

7.25 SKIP

Syntax: skip [address],[condition] [index_0] [index_1]

This pseudo-instruction causes the program to skip to [address]. If you do not specify [address], the program skips to the ENDTEST pseudo-instruction. If you specify [condition], it may be any of three types. If the [condition] is ONERROR, the program skips only when the error flag is set. If the [condition] is NOERROR, the program skips only when the error flag is not set. If the [condition] is ONEQUAL, [index_0] and [index_1] are also specified. The SKIP pseudo-instruction compares them and skips if they are equal.

7.26 END TEST

Syntax: ENDTEST

This pseudo-instruction terminates each test. It tests various flags and acts accordingly.

If you type in DI TE:<test-number>, the monitor loops on the test. Otherwise, the instruction continues the program by loading the next test overlay from the TU58.

7.27 ERRLOG

Syntax: ERRLOG

This pseudo-instruction logs in DCS up to 64 single-bit main memory errors.

7.28 DUMPLOG

This pseudo-instruction types out the contents of the errors log.

BLANK

CHAPTER 8
MICRODIAGNOSTIC PROGRAM LISTINGS

8.1 SCOPE

Microdiagnostic program error messages usually provide enough information to isolate and repair faults in the processor. Error messages list the failing test number and call out suspected components and modules. However, if the specified parts are changed and the same message prints after rerunning the program, the program listing provides a further troubleshooting resource. Using the microfiche card for the appropriate set of tests, locate the frame for the failing test through the index in the lower right microfiche frame.

8.2 TEST OVERLAY LISTING FORMAT

Each test overlay consists of a test and associated data. A test may consist of several subtests. Although test overlays vary slightly, they use the same general format.

```
Test title
Documentation
        Functional description
        Test algorithm
        Test patterns
        Logic description
        Assumptions
        Error description
Subtest 1
        Pseudo-instructions                    ; comments
Subtest 2...
        Pseudo-instructions                    ; comments
Test data
        8085 memory buffer data                ; comments
        DCS data                               ; comments
The monitor loads this directly into DCS.

        RTEMP data      ; Data loaded beginning at RTEMP0.
        Cache data      ; Data loaded beginning at cache
                        ; address 0.
```

NOTE

Before analyzing the test code and data, read the documentation carefully. It gives an overall description of the test and many useful details.

8.3 TEST EXECUTION FLOW

The execution flow for each subtest in a test consists of three steps.

1. Pseudo-instructions set up test data and parameters. The 8085 processor performs these functions.

2. Microcode in DCS executes.

3. Pseudo-instructions check the results of microcode execution. If the check reveals no error, control passes to the next subtest or test. Flag settings, error conditions, and test structure cause the program to loop to the beginning of DCS, or to a loop point in the pseudo-instructions, as appropriate.

Consider the test overlay listing in the following example.

I. Functional Description

This test consists of two subtests.

1. Cache error register
2. Cache control register

These are read/write registers in the CAK chip.

II. Test Algorithm

1. Modify DCS address 1 to select the appropriate temporary register for the subtest.

2. Set up a loop to select test patterns.

3. Load WD register (DM) with test pattern.

4. Execute DCS program.

   a. Load memory status and control address register (MEMSCAR) with address in temporary register.

   b. Write test pattern in WDREG (DM) to memory status and control register (MEMSCR).

   c. Read test pattern back to WHREG (DM).

5. Compare results (expected and received).

6. If no error, go to step III for remaining test patterns.

III. Test Patterns

Iteration	Test Pattern
1	A
2	5
3	3

IV. Logic Description

This test checks only the S/C registers that reside in the CAK gate array. The registers are accessed via WBUS<27:24>. If on error, replacing the CAK does not work, try running the MEMSCAR test, which checks for a multiple addressing problem. If that fails, then one of the other gate arrays with S/C registers is causing trouble.

V. Assumptions

This test assumes that the W-BUS is operational.

VI. Error Description

EXPECTED MEMSCR DATA
RECEIVED MEMSCR DATA
LOOP COUNT

```
0004  472   ;SUBTEST 1 Cache Error Register
0004  476   SUBTEST                          ;  Cache error reg.
0006  477   INITIALIZE                       ;  Initialize CPU.
0008  478   LOADDCS     4$,,01,1             ;  Modify DCS address 01.
000F  479   BEGINSA                          ;  Signature analyzer loop
                                            ;  point.
0011  480   LOOP        I,1,3                ;  Create test loop.
0018  481   LOADREG     WDREG,1$,I,L         ;  Test pattern to WD reg.
0020  482   ERRLOOP                         ;  Error loop.
0022  483   FETCH      0                    ;  Start DCS program.
0026  484   BRSTCLK     4                    ;  End DCS program.
002A  485   CMPREGMSK  WHREG,1$,I,3$,,L,    ;  Compare results.
0036  486   IFERROR     ,<<CAK>>,            ;  Possible failing array.
003C  487   ENDLOOP     I                    ;  End test loop.
003E  488   ENDSA                           ;  End signature analyzer
                                            ;  loop.
0041  491   ;SUBTEST 2 Cache Control Register

0041  495   SUBTEST                          ;  Cache control reg.
0043  496   INITIALIZE                       ;  Initialize CPU.
0045  497   LOADDCS     2$,,01,1             ;  Change DCS address 01.
004C  498   BEGINSA                          ;  Signature analyzer loop
                                               point.
004E  499   LOOP        I,1,3                ;  Create test loop.
0055  500   LOADREG     WDREG,1$,I,L         ;  Test pattern to WD reg.
005C  501   ERRLOOP                         ;  Error loop.
005F  502   FETCH      0                    ;  Start DCS program.
0063  503   BRSTCLK     4                    ;  End DCS program.
0067  504   CMPREGMSK  WHREG,1$,I,3$,,L,    ;  Compare results.
0073  505   IFERROR     ,<<CAK>>,            ;  Possible failing array.
0079  506   ENDLOOP     I                    ;  End test loop.
007C  507   ENDSA                           ;  End signature analyzer
                                            ;  loop.
007E  508   SKIP                            ;  SKIP to line 568.


0084  513                BEGIN TEST DATA
            ;--------------------------
0084  515   1$:    .LONG          ^X0A000000      ;  Test pattern.
0088  516         .LONG          ^X05000000      ;  Test pattern.
0086  517         .LONG          ^X03000000      ;  Test pattern.
008E  518   2$:
008E  519   ;%  01  MEMSCAR R[Temp1]            ;  Rtemp for subtest
                                                ;  2.
009B  523   3$:    .LONG          ^X0F000000      ;  Mask for CMPREGMSK
                                                ;  pseudo op.
```

```
009F   524  4$:
009F   525  ;%   01    MEMSCAR R[Temp0]                    ;  Rtemp for subtest
                                                          ;  1.

00AA   532             BEGIN_CPU_DATA

0002   534             DCS_DATA

0001   536  ;%   00    NOP
000C   540  ;%   01    MEMSCAR R[TEMP0]                    ;  Write add of S/C to
                                                          ;  MEMSCAR.
0017   544  ;%   02    WB_RDM,WCTRL/MEMSCR_WB              ;  Write test pattern
                                                          ;  to MEMSCR.
0022   548  ;%   03    RDM_WB,WCTRL/MEMSCR                 ;  Read test pattern
                                                          ;  back to RDM.
002D   552  ;%   04    NOP,STOP.CLOCK                      ;  Stop CPU clock.

0038   557             RTEMP_DATA

0001   559        .LONG              ^X04000000          ;  Cache error
                                                          ;  register address.
0005   560             .LONG         ^X06000000          ;  Cache control
                                                          ;  Register address.

0009   562             END CPU DATA
00A2   566             END TEST DATA

               ;-----------------------------
00A2   568                   ENDTEST
```

Figure 8-1 illustrates the execution flow for the first subtest.

Note that FETCH gets control of the control store address lines. Then the microcode executes during BRSTCLK. The BRSTCLK pseudo- instruction checks that the CPU clock stops at the DCS address specified (4). BRSTCLK then passes control to the next pseudo- instruction.

8.4 MICROINSTRUCTION REFERENCES

For detailed information on the microassembler, see the *MICRO2 User's Guide Reference Manual* (AA-H531A-TE).

For accurate interpretation of micro-orders, see the machine definitions (micro-order definitions) and the macro definitions in the VAX-11/750 microcode listings. For special macros used only in microdiagnostics, see the microdiagnostic listings.

8.5 MICROINSTRUCTION INTERPRETATION TIPS

The DCS data in the listings consists of two types of statements: micro-orders and microinstruction macros. Micro-orders take the form <field>/<value>, where <field> is a field of bits in the control store word and <value> is a hexadecimal number to be placed in that field. For example, WCTRL/MEMSCR (from line 548 of the example in Paragraph 8.3) puts the value identified by MEMSCR into the WCTRL field, bits <30:25>, of the control store word.

However, to find out what the WCTRL field actually does with that value, the machine definitions section of the VAX-11/750 microcode listings under WCTRL must be checked. The comment in the listing for the WCTRL/MEMSCR micro-order reads as follows.

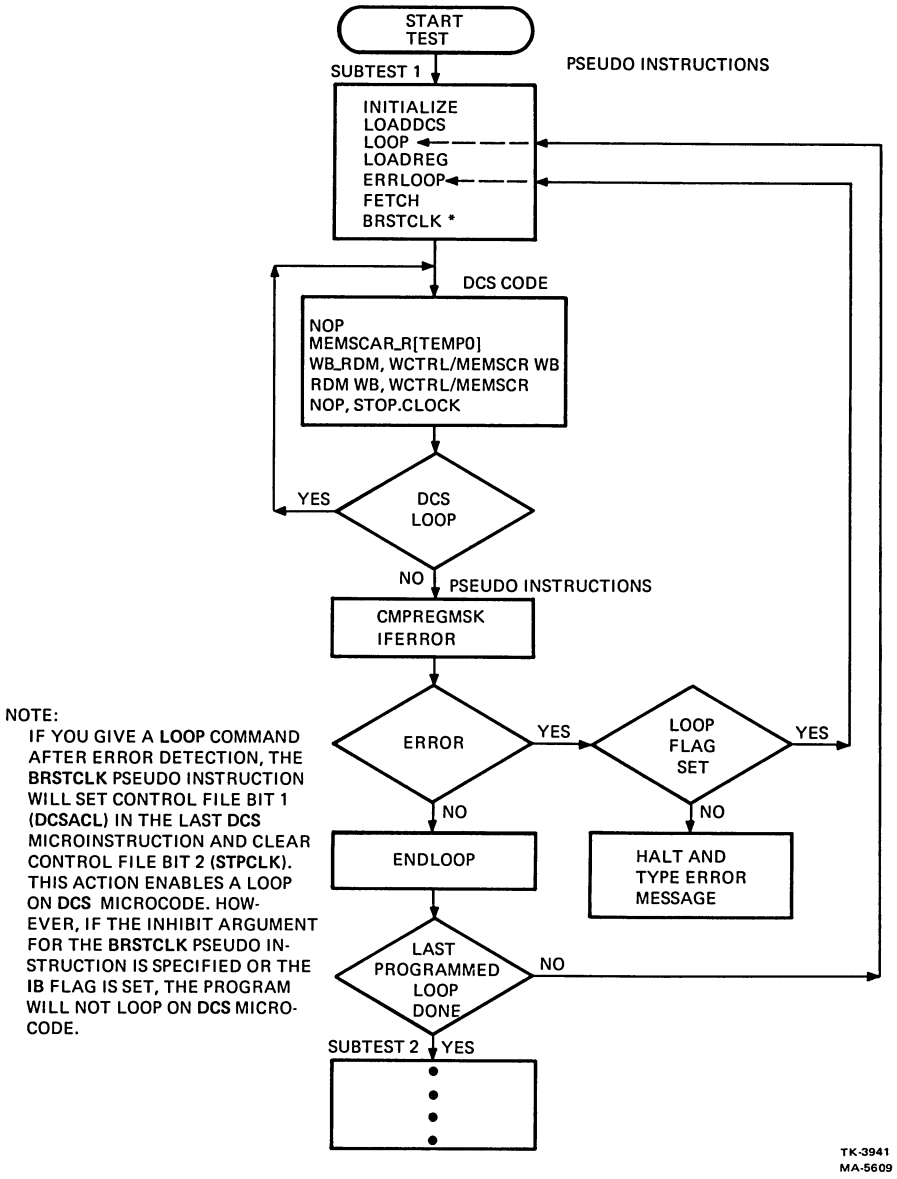


Figure 8-1 Sample Subtest Execution Flow

;WBUS<27:24><– Memory Status and Control Reg (@MEMSCAR)

The micro-order therefore asserts bits <27:24> from the memory status and control register, addressed by MEMSCAR, on the W-BUS.

Macros generate the microcode for one or more micro-orders within one microinstruction. Generally, the function of a macro can be interrupted by analyzing the macro name. Macros like NOP and STOP.CLOCK are obvious.

All macros that include an underscore symbol are register transfer macros. The transfer goes from the right term to the left term. However, when the underscore symbol is part of a micro-order, it does not indicate the presence of a macro within the micro-order.

Commas separate micro-orders and macros, when several are used to form one microinstruction.

Consider the DCS data in the example in Paragraph 8.3.

Line 540: MEMSCAR R[TEMP0]

Move the contents of the scratch pad register RTEMP0 to MEMSCAR, the memory status and control address register.

Line 544: WB RDM,WCRTL/MEMSCR WB

The WB RDM macro moves data from the WDREG on the DM to the W-BUS. WCTRL/MEMSCR WB, according to the micro-order definition in the VAX-11/750 microcode listing, moves W-BUS bits <27:24> to the memory status and control register. This is a write function.

Line 548: RDM WB,WCTRL/MEMSCR

WCTRL/MEMSCR moves the data from MEMSCR to the W-BUS bits <27:24>. The macro RDM WB moves that data from the W-BUS back to the WHREG on the DM. This is a read function.

8.6 DIAGNOSTIC CONTROL STORE AND CONTROL FILE

The diagnostic control store (DCS) contains 64 locations that are each 88 bits wide. The DCS address space runs from control store address 1800 to 183F. The first 80 bits in each word, bits <79:00>, correspond to the 80 bits of the VAX-11/750 microword. The upper eight bits in each word, bits <95:88>, make up the control file. They enable and disable control functions on the DM. The intervening bit numbers, bits <87:80>, are reserved for the macro assembler.

The microdiagnostic tests control the eight bits of the control file with eight microinstruction macros (Table 8-1). These macro functions correspond to the set control file functions listed in Paragraph 6.8.

8.7 LOOPING AND STEPPING THROUGH TESTS

After analyzing the test description, code, and comments for the failing test, you may want to analyze certain signals in the processor. Locate points of interest in the circuit under test in the VAX-11/750 print set.

To loop on the failing test at machine speed, follow the command sequence shown below.

```
DM>TE/C<RET>                          ! Load the microdiagnostic
                                      ! monitor.
MIC>DI TE:4<RET>                      ! Execute the failing
                                      ! test.
ECKAA-VO.22 MIC (L0003)-V00.04
04,
?ERROR:  0036 TEST: 04 SUBTEST: 01

DATA:    0A000000                     ! Expected cache error register
                                      ! contents.
         08000000                     ! Received cache error register
                                      ! contents.
         00000001                     ! Loop count.
                                      ! See Error Description, example
                                      ! in Paragraph 8-3.
Failing Gate Arrays: CAK,
MIC>LO<RET>                           ! Loop on test 4.
```

Table 8-1 DM Control File Macros

Macro Name	Bit	DMCF	Function
STROBE.V.BUS	88	0	Strobes VBus at rising edge of next M clock.
DCS.ADDR__0	89	1	Clears DCS address register. Returns to DCS address 0.
STOP.CLOCK	90	2	Stops CPU clock.
LATCH.CS.ADDR	91	3	Latches trace register at rising edge of next M clock.
RDM WB	92	4	Reads data on W-BUS into WHREG on DM at rising edge of next M clock.
WBRDM	93	5	Asserts (writes) contents of WDREG (on DM) on W-BUS for entire microcycle.
DM CMI	94	6	Reads data on CMI bus into the MHREG on DM at the rising edge of next M clock.
INH.CLOCK.SA	95	7	Inhibits clocking signature analyzer on M clock.

To step through the entire test, one pseudo-instruction at a time, follow the steps shown in the example below. Note the flow of the subtest sequence in the TPC addresses.

```
MIC>DI TE:4<RET>                          ! Execute the failing test.
ECKAA-VO.22 MIC (L0003)-V00.04
04,
?ERROR:  0036 TEST: 04 SUBTEST: 01

DATA:    0A000000                         ! Expected data.
         08000000                         ! Received data.
         00000001                         ! Loop count.
Failing Gate Arrays: CAK,
MIC>SE ST IN                              ! Set step instruction.

MIC>CO<RET>
         TPC= 003C                        ! Test continues at this point.
         TPC= 0018                        ! Restart subtest 1, using
         TPC= 0020                        ! the second test pattern, I = 2.
         TPC= 0022
         TPC= 0026
         TPC= 002A
         TPC= 0036
```

```
            TPC= 003C
            TPC= 0018                   ! Restart subtest 1, using
            TPC= 0020                   ! the third test pattern, I = 3.
            TPC= 0022
            TPC= 0026
            TPC= 002A
            TPC= 0036
            TPC= 003C
            TPC= 003E
            TPC= 0041                   ! Start subtest 2, using
            TPC= 0043                   ! the first test pattern, I = 1.
            TPC= 0045
            TPC= 004C
            TPC= 004E
            TPC= 0055
            TPC= 005C
            TPC= 005F
            TPC= 0063
            TPC= 0067
            TPC= 0073
            TPC= 0079
            TPC= 0055                   ! Restart subtest 2, using
            TPC= 005C                   ! the second test pattern, I = 2.
            TPC= 005F
            TPC= 0063
            TPC= 0067
            TPC= 0073
            TPC= 0079
            TPC= 0055                   ! Restart subtest 2, using
            TPC= 005C                   ! the third test pattern, I = 3.
            TPC= 005F
            TPC= 0063
            TPC= 0067
            TPC= 0073
            TPC= 0079
            TPC= 007C
            TPC= 007E
            TPC= 00AA
            TPC= 0004                   ! Loop to restart subtest 1,
            TPC= 0006                   ! using the first test pattern,
            TPC= 0008                   ! I = 1.
CTRL/C      TPC= 000F↵C                 ! Type CTRL/C (or carriage
                                        ! return) to return to the
                                        ! microdiagnostic monitor.
MIC>
```

To step through the DCS code with a specific pattern in a specific subtest, step through the pseudo-instructions to the proper point. Then type SE ST CY (set step cycle) to stop at each DCS instruction. The example below shows how to step through the DCS code, using test pattern 2 in subtest 1.

```
MIC>DI TE: 4<RET>                       ! Execute the failing test.
ECKAA-VO.22 MIC (L0003)-V00.04
04,
?ERROR: 0034 TEST: 04 SUBTEST: 01
```

```
DATA:    0A000000          ! Expected data.
         08000000          ! Received data.
         00000001          ! Loop count.
Failing Gate Arrays: CAK,
MIC>SE ST IN<RET>          ! Set step instruction.

MIC>CO<RET>

         TPC= 003C
         TPC= 0018         ! Restart subtest 1, using
         TPC= 0020         ! the second test pattern.
         <RET>             ! Carriage return.

MIC>SE ST CY<RET>

MIC>CO<RET>                ! Step through DCS
                           ! with the second test pattern.

         DCS ADDR= 00
         DCS ADDR= 01
         DCS ADDR= 02
         DCS ADDR= 03
         DCS ADDR= 04
MIC>
```

Note that control returns to the microdiagnostic monitor automatically, after execution of the last micro instruction at DCS ADDR= 04.

Example 4 shows how you can force a loop in the DCS code with the Set Control File command. Set control file bit 1 and clear bit 2 in the last DCS word in the sequence.

Example 4

```
MIC>SE CF:4 1<RET>         ! Set CF bit 1 in DCS
                           ! address 4.
MIC>CL CF:4 2<RET>         ! Clear CF bit 2 in DCS
                           ! address 4.
MIC>CO<RET>                ! Continue.
```

BLANK

APPENDIX A
CABLE CONNECTIONS AND TEST POINTS

Table A-1 lists the backplane pin numbers for the local terminal, the TU58, and the front panel cables. Table A-2 lists backplane pin numbers for specific DM test points.

Table A-1 Backplane Pins for Local Terminal, TU58, and Front Panel Cables

Local Terminal	
Pin	**Signal Name**
C0624	Ground
C0634	DM terminal serial out
C0632	DM terminal serial in
C0645	Console baud rate A L
C0646	Console baud rate B L
C0649	Console baud rate C L
C0650	Console baud rate D L
TU58	
Pin	**Signal Name**
B0686	TU58 serial out (signals between TU58 and DM)
B0688	TU58 serial in (signals between TU58 and DM)
B0685	EIA TU serial out L (signals between DM and CPU)
B0687	EIA TU serial in L (signals between DM and CPU)
Front Panel	
Front panel cable is plugged into SET A (top half), with cable pin 1 on top.	

Table A-2 DM-Specific Backplane Pins

Pin	Signal Name
C0673	DM SA CLK L (D)
C0675	SA ST/SP L (D)
C0674	DM RESET L (R)
C0681	DM MATCH PULSE (D)

D=Drive
R=Receive

APPENDIX B
DM SPECIFICATIONS

Size: 31.12 cm (12.25 in) × 40.01 cm (15.75 in) extended hex	
Pin Configuration:	3 × 94 — Standard L series
Power:	
Maximum	+5 V @ 12.9 A, +12 V @ 120 mA, −15 V @ 85 mA
Typical	+5 V @ 9.6 A, +12 V @ 60 mA, −15 V @ 30 mA
Technology:	TTL (Schottky, Low Power Schottky)
	MOS

Environmental standards (temperature, pressure, etc.) are the same as for the VAX 11/750.

BLANK

APPENDIX C
BOOTING THE DIAGNOSTIC SUPERVISOR FROM THE TU58

If the diagnostic supervisor (ECSAA.EXE) cannot be booted from the system disk, it may be booted or loaded from the TU58 tape unit. The RD diagnostic tape in the VAX-11/750 Microdiagnostic Tape Kit contains the diagnostic supervisor. The tape also contains two diagnostics (that run under the supervisor) and the CPU hardcore tests. Table C-1 lists the programs on the RD diagnostic tape, and indicates which can be booted, as well as the load and start addresses.

Table C-1 RD Diagnostic Tape Programs

Diagnostic Name	Description	Bootable	Load Address	Start Address
ECSAA.EXE	Diagnostic supervisor	Yes	FE00	10,000
ECCBA.EXE	UBI Diagnostic	N/A	N/A	N/A
ECKAX.EXE	CPU cluster exerciser	N/A	N/A	N/A
EVKAA.EXE	CPU hardcore tests	No	0	200

The following dialogues show how to boot or load the diagnostic supervisor and the CPU hardcore tests off the RD diagnostic tape.

Booting the diagnostic supervisor:

```
>>>B DDA0<RET>

DIAGNOSTIC SUPERVISOR. ZZ-ECSAA- - - - -

DS>
```

Loading and running the diagnostic supervisor:

```
>>>>CSRL/D

RDM> LO ECSAA.EXE FE00<RET>

RDM> RET/D<RET>

>>>S 10000<RET>

DIAGNOSTIC SUPERVISOR. ZZ-ECSAA- - - - -
```

Loading and running the CPU hardcore tests:

>>>**CTRL/D**

RDM> **LO EVKAA.EXE<RET>**

RDM> **RET/D<RET>**

>>>**S 200<RET>**

EVKAA-5.0 done!
EVKAA-5.0 done!

BLANK